VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Diplomová práce

2011 Bc. Pavel Bednář

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro práci s Office Open XML v OS Android.

Application for Office Open XML in OS Android

2011 Bc. Pavel Bednář

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Pavel Bednář**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Aplikace pro práci s Office Open Xml v OS Android
Application for Office Open XML in OS Android

Zásady pro vypracování:

Jelikož operační systém Google Android je poměrně nový, neexistuje pro něj takové množství aplikací jako třeba pro Microsoft Windows Mobile. Jednou z nepostradatelných aplikací každého systému by měl být balík pro práci s dokumenty. Těch však v androidu není dostatek a většinou umí dokument pouze otevřít. Pokud uživatel potřebuje vytvářet dokumenty, musí sáhnout po placené verzi, kterých i tak není dostatek. Hlavním cílem práce je vytvořit je srovnat již existující balíky. Srovnat jejich výhody a nevýhody, porozumět práci s otevřenými formáty dokumentů OpenXML. Na základě těchto poznatků pak navrhnout a implementovat vlastní aplikaci, která by umožňovala nejen čtení ale i vytváření a editaci dokumentů OpenXML.
Práce bude obsahovat:
1. Porovnání již existujících balíků. Zjištění jejich výhod a nedostatků.
2. Analýza OpenXml.
3. Návrh implementace vlastního kancelářského balíku.
4. Implementace kancelářského balíku - prvního prototypu se základní funkčností čtení a zápisu "Word" a "Excel" dokumentů.
5. Zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Svatopluk Štolfa, Ph.D.**

Datum zadání: 19.11.2010
Datum odevzdání: 06.05.2011

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 4. 5. 2011

Podpis

Poděkování

Rád bych poděkoval vedoucímu diplomové ing. Svatoplukovi Štolfovi za odbornou pomoc a konzultace při vytváření této práce.

Abstrakt

Formát Open Office XML je díky své desktopové implemenci v kancelářském balíku Microsoft Office 2007 velice rozšířeným formátem pro práci s dokumenty. Svou podobu najde i na chytrých mobilních zařízeních, které se stávají nedílnou součástí našeho života. Díky rozvoji nových trendů dnes mobilní zařízení disponují velkým displejem, který přináší nové možnosti pro aplikace. Dříve nebo později se musely objevit i aplikace pro práci s dokumenty. A proč by přinášeli nové formáty, když můžou použít standartizované formáty, které zaručí, že takto vytvořený dokument půjde otevřít i na jiném zařízení nebo operačním systému. Nejrozšířenější standartem, které tyto aplikace podporují, je právě Office Open XML. Většina těchto aplikací umožňuje pouze zobrazení dokumentu. Pokud chcete dokumenty vytvářet, musíte využít placené verze. Cílem této práce je porozumět standardu Office Open XML, zejména těm částem, týkajících se textového editoru a tabulkového procesoru. Srovnat existující dostupné aplikace a na základě získaných zkušeností navrhnout svou vlastní implementaci kancelářského balíku.

Klíčová slova

Office Open Xml, Android, WordProcessingML, SpreadsheetML

Abstract

Open Office XML format, thanks to its desktop implementation Microsoft Office 2007 is very widespread format for working with documents. Its implementations can be found on smart mobile devices, which which are becoming an integral part of our lives. Thanks to the development of new trends in mobile devices todays devices have a large display that brings up new possibilities for applications. Sooner or later applications for working with documents had to apper. And why would bring a new format when they can use standardized formats to ensure that their document can be opened on another device or even the operating system. The most common standards that uses these applications is currently the Office Open XML. Most of these applications allow you to only view the document. If you want create documents you have to reach out for the paid version. The goal of this work is understanding of the Office Open XML especially word processing and spreadsheets. Make comparison of existing applications available and based on experience design my own office suite implementation.

Key words

Office Open Xml, Android, WordProcessingML, SpreadsheetML

Seznam použitých symbolů a zkratek

ADT	Android Devepoment Toolkit
AVD	Android Virtual Device
ECMA	European Computer Manufacturers Associatio
GPS	Global Positioning Systém
HTML	HyperText Markup Language
IEC	Internacionál Elektrotechnical Commission
IDE	Integrated Development Environment
ISO	International Organization for Standartization
JDK	Java Development Kit
JRE	Java Runtime Environment
MSDN	Microsoft Documentanion Network
OHA	Open headset Aliance
OOXML	Office Open Office
OPC	Open Packaging Conventions
OS	Operační systém
PC	Personal Computer
PDA	Personal Digital Assistant
SDK	Software Development Kit
VM	Virtual Machine
XML	Extensible Markup Language
XSD	XML Schema Definition

Seznam obrázků

Seznam výpisů zdrojového kódu

Obsah

1 Úvod

Android je v současnosti jedním z předních operačních systémů pro mobilní zařízení. Softwarová základna pro Android je již poměrně rozsáhlá a obsahuje nepřeberné množství aplikací. Jednou ze základních kategorií jsou i kancelářské aplikace. V této kategorii již existuje spousta aplikací, mezi kterými dominuje zejména formát Office Open XML. Avšak většina těchto aplikací neumožňuje zdarma vytvářet nebo editovat dokumenty. Tato práce se zabývá možnostmi tohoto formátu v prostředí Android. Nejprve popisuji základní strukturu Open XML, jak jsou organizovány jazyky WordProcessingML a SpreadSheetML, které slouží k ukládání dokumentů pro Microsoft Word a Microsoft Excel. V další části práce srovnávám existující kancelářské balíky a navrhhuji vlastní aplikaci, která umí uložit nebo otevřít dokument WordProcessingML a sešit SpreadSheetML.

2 Operační system Android

2.1 Co je operační systém Android

Je to platforma společnosti Google, která založená na linuxovém jádře a je určena pro mobilní telefony, PDA a tablety. Je to otevřený systém, do kterého lze psát aplikace jednoduše v Javě, s pomocí knihoven poskytnutých od Google. Android se v současné době stává stále populárnějším a stále více výrobců distribuuje své telefony s OS Android. Mezi nejvýznamnější výrobce patří například: HTC, Samsung a LG. V poslední době se začíná Android objevovat i na tabletech nebo elektronických čtečkách knih.

2.2 Historie

Platforma Google Android byla ohlášena 5. listopadu 2007 a současně bylo založeno i konsorcium sdružující výrobce hardware a software a telekomunikačních společností Open Headset Alliance (OHA). To se podílí na prosazování otevřených standardů ve světě mobilních zařízení. O týden později 12. listopadu 2009 byla představena první verze SDK, která obsahovala emulátor, debugger, knihovny, dokumentaci, ukázkové programy a tutoriály. SDK si mohl nainstalovat a vyzkoušet každý uživatel operačního systému Windows XP, ale i novějších Mac OS X nebo Linux. K vývoji bylo nutné použít Eclipse s nainstalovaným rozšířením ADT (android developer tools)

První verze operačního systému spatřila světlo světa 23. září 2008. U nás se první telefony s Androidem 1.0 objevily až v lednu 2009 u společnosti T-Mobile.

2.3 Přehled verzí

Cupcake (1.5)
30. dubna 2009 se objevil formou aktualizace. Aktualizace na Android 1.5 (Cupcake) přidává několik nových funkcí a vylepšení:

- možnost nahrávat a sledovat videa z kamery
- nahrávání videí na YouTube a fotografií na Picasu přímo z telefonu
- nová softwarová klávesnice s automatickým dokončováním slov.
- bluetooth – podpora A2DP
- možnost automatického připojení Bluetooth headset
- nové widgety a složky
- animace při přechodu mezi obrazovkami
- rozšíření funkce kopírovat a vložit.

Obrázek 1 Cupcake

Donut (1.6)

Další verze byla uvedena jen pár měsíců po Cupcake, 15. září 2009 se jménem Donut (kobliha). Hlavní novinkou byla syntéza řeči a podpora zařízení s WVGA rozlišením. Změny:

Obrázek 2 Donut

- vylepšený Android Market
- nové prostředí fotoaparátu, kamery a galerie
- galerie umožňuje označit více fotografií k vymazání
- aktualizované vyhledávání hlasem
- Quick Search Box – umožňuje vyhledávat záložky, historii, kontakty a na webu z domovské obrazovky
- podpora pro technologie CDMA/EV-DO, 802.1x, VPN, Gesta a syntéza řeči
- podpora pro WVGA rozlišení displeje
- vylepšení rychlosti vyhledávání a kamery

Eclair (2.0 a 2.1)

Označení Eclair obsahuje dvě rozdílné verze. První verze 2.0 byla světu představena 26. října 2009, Druhá SDK verze 2.1 pak 12. ledna 2010. Hlavními přínosy byly zejména podpora HTML5 a aktivní tapety na ploše.

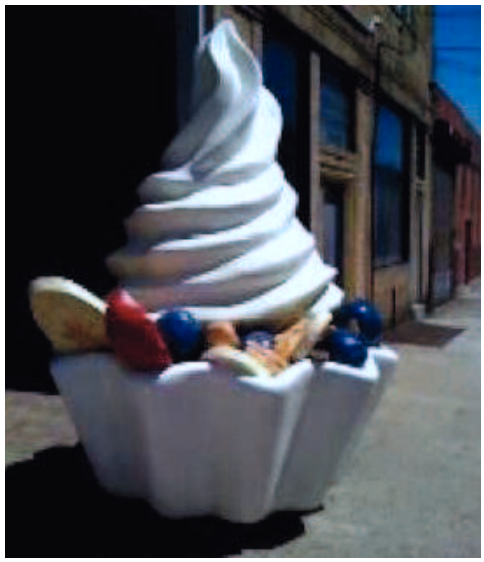
Obrázek 3 Eclair

- optimalizována rychlost hardwaru
- podpora pro více velikostí a rozlišení displeje
- zdokonalené uživatelské prostředí
- nové prostředí prohlížeče a podpora HTML5
- nový seznam kontaktů
- mapy Google aktualizovány na 3.1.2
- podpora pro Microsoft Exchange
- podpora přisvětlovací diody
- digitální zoom (fotoaparát)
- vylepšená softwarová klávesnice
- podpora pro Bluetooth 2.1
- animované tapety na domovské stránce.

Froyo (2.2)

Krycí jméno Froyo je opět „zákusek“ – FROzen YOgurt. Byl představen 20. května 2010 na konferenci Google I/O. K uživatelskému prostředí přidává nové technologie a funkce:

- možnost instalovat aplikace na paměťovou kartu
- Adobe vydalo plugin Adobe Flash 10.1, není integrován do systému, distribuce je řešena přes Android Market nebo přes stránky Adobe
- díky JIT (Just-in-time) kompilátoru se podařilo zvýšit rychlost systému na různých benchmarcích 2x až 5x, dále je vylepšena správa paměti RAM
- možnost vytvořit z telefonu WiFi hotspot, nebo sdílet internetové připojení přes USB kabel
- dva nové režimy telefonu – „car mode“ a „night mode“ (režim v autě a noční režim)
- více nastavení fotoaparátu a kamery
- přidána podpora pro OpenGL ES 2.0, vícebarevný trackball, vylepšena podpora pro Exchange, Bluetooth a přidána další vrstva vývojářského API



Obrázek 4 Froyo

Gingerbread (2.3 a 2.4)

Dne 6. prosince 2010 byla vypuštěna verze Android 2.3 Gingerbread s podporou video formátu WebM a NFC. Verze 2.4 zatím nebyla představena, její hlavní změnou bude podpora 2 jádrových procesorů:

Obrázek 5 Gingerbread

- podpora video formátu WebM pro HTML5 video[11]
- podpora pro Near Field Communication standard, který dnes podporují některé mobilní telefony
- podpora SIP protokolu pro internetovou telefonii
- lepší správa prostředků
- upravená virtuální klávesnice
- vylepšená funkce kopírovat a vložit[12]
- podpora více kamer a nových sensorů
- nové Google Maps 5 s 3D přístupem (Dostupné přes market pro všechny verze androida)
- rozšíření podpory nativního kódu.

Honeycomb (3.0)

Je to revoluční verze, která je orientována hlavně na tablety a zařízení s větším rozlišením displeje. SDK bylo představeno 24. února 2011. Přináší vylepšený multitasking a nové 3D prostředí:

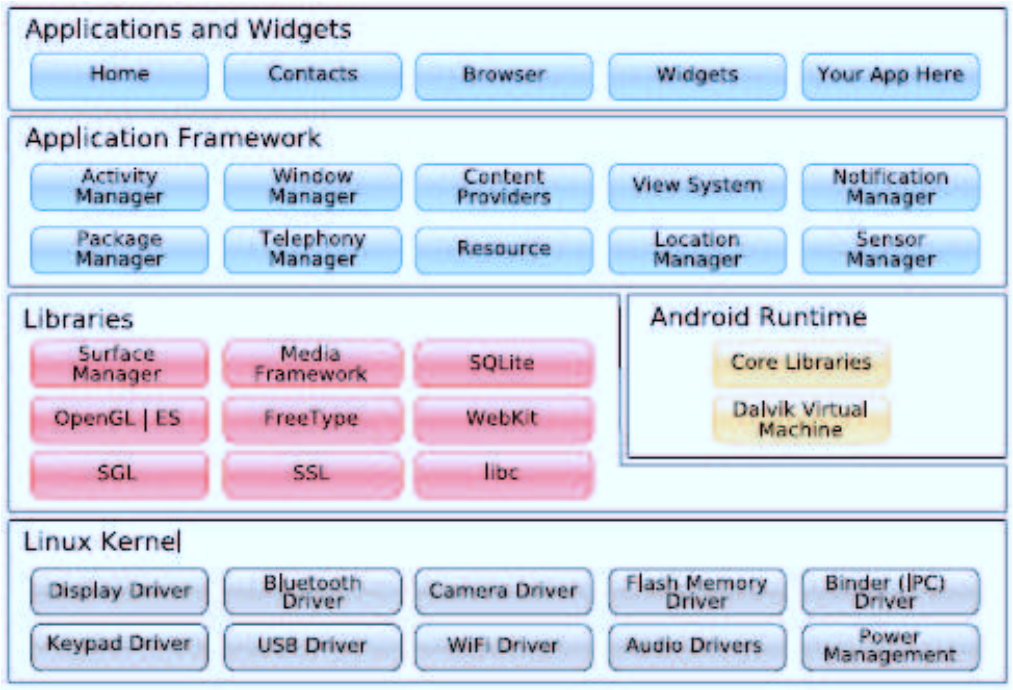
- vylepšený multitasking
- nový systém notifikací
- upravena hlavní plocha a widgety
- knihovna Google eBooks
- google voice pro přenos video a audio hovorů.

Obrázek 6 Honeycomb

2.4 Architektura Android

Nyní, když máte představu o tom, co je to Android, pojďme se podívat, jak vlastně funguje. Některé části OS Android jsou všeobecně známé, jsou to: linuxové jádro, OpenGL a SQL databáze. Jiné pojmy jsou zcela nové, jako například životní cyklus Android aplikací.

Je třeba být dobře seznámen s těmito pojmy, abyste byli schopni napsat kvalitní aplikaci pro Android. Architektura systému se skládá z klíčových vrstev a součástí, které tvoří Android. Na obrázku 7 můžete vidět základní schéma architektury Android. Každá vrstva využívá služeb vrstev pod ní.



Obrázek 7 Architektura systému Android

2.4.1 Linuxové jádro

Android je postaven na osvědčeném základu linuxového jádra. Vytvořil ho Linus Torvalds v roce 1991. Linux lze dnes najít ve velkém množství zařízení, od náramkových hodinek až po superpočítače. Linux poskytuje abstraktní hardwarovou vrstvu pro Android, která umožňuje to, že bude moci být Android v budoucnosti přenesen na celou řadu platforem.

Android používá Linux pro jeho správu paměti, správu procesů a další služby operačního systému. Uživatel systému Android nikdy neuvidí Linux přímo, a vaše programy nebudou provádět přímá volání do jádra. Jako vývojář, si ale musíte být vědom, že tam je.

Avšak některé nástroje, které budete potřebovat v průběhu vývoje, komunikují s Linuxem. Například příkazový řádek ADB Shell, který otevře okno s příkazovým řádkem Linuxu, můžete procházet přímo adresářovou strukturu Linuxu a provádět příkazy pro úpravu systému nebo třeba zobrazit aktivní procesy.

2.4.2 Knihovny

Další vrstvou v pořadí jsou knihovny (Native Libraries). Tyto knihovny jsou většinou psané v jazyce C nebo C++. Jsou kompilovány a dodávány přímo výrobcem pro danou architekturu zařízení.

2.4.2.1 Surface Manager

Je to správce oken, podobně jako Windows Vista nebo Compiz v Linuxu, ale je mnohem jednodušší. Namísto kreslení přímo do vyrovnávací paměti obrazovky, příkazy pro vykreslení jdou do bitmap mimo obrazovku, kde jsou kombinovány s jinými bitmapami a vytvoří tak obraz, který uvidí uživatel. To umožní systému provádět spoustu grafických efektů.

2.4.2.2 2D a 3D grafika

Plošinové a prostorové elementy mohou být snadno kombinovány do jednoho uživatelského rozhraní. Pokud zařízení disponuje podporou 3D akcelerace, knihovna jí využije. Pokud ne, použije se místo něj rychlý softwarový renderer.

2.4.2.3 SQL Databáze

Android zahrnuje odlehčenou verzi SQLite databázového engine. SQLite je sice malý, ale za to výkonný databázový engine. Vytvořil jej Dr. Richard Hipp v roce 2000. Je to jeden z nejvíce rozšířených databázových engine na celém světě. Lze jej mimo jiné najít i v Apple iOS, symbian telefonech, OS X a mnoha dalších.

2.4.3 Android runtime

Další vrstvou nad linuxovým jádrem je Android Runtime včetně virtuálního stroje Dalvik s knihovnami jazyka Java. Dalvik je virtuální stroj (VM), který navrhl a napsal Dan Bornstein z Google. Váš kód se zkompiluje do strojově nezávislého kódu nazývaného bytecode. Ten je pak virtuálním strojem Dalvik spuštěn v mobilním zařízení. Ačkoliv bytecode Java a Dalvik jsou malinko rozdílné, v podstatě je Dalvik pořád virtuální stroj Java, akorát je optimalizován na nízké paměťové požadavky. Umožňuje více instancí VM a využívá linuxové jádro pro bezpečnost a izolaci procesů.

2.4.4 Aplikační Framework

Tato vrstva poskytuje základní stavební bloky, které využijete při tvorbě aplikací. Framework je zabudován přímo do Android, ale dá se rozšířit pomocí vlastních komponent. Důležitou součástí Framework je Activity Manager, který kontroluje životní cyklus aplikací. Notification Manager je zodpovědný za události, jako jsou příchozí zprávy, oznámení nebo zprávy ze senzorů.

2.4.5 Aplikace a widgety

Jsou nejvyšší vrstvou v architektuře Android. Obsahuje aplikace, které uvidí běžný uživatel, který není seznámen s architekturou Android. Aplikace jsou programy zobrazené přes celou obrazovku, s nimiž uživatel pracuje. Na druhou stranu widget pracuje pouze v malém prostoru na domácí ploše.

2.5 Životní cyklus Android aplikací

V operačních systémech jako jsou Windows nebo Linux jste zvyklí mít mnoho běžících a viditelných oken současně a jen jedno z těchto oken je aktivní (má fokus). Android, ale pracuje jinak. V OS Android je pouze jedna aplikace, která je v popředí, a zabírá celou obrazovku pro sebe, kromě notifikační lišty.

Pokud spustíte aplikaci, Android ji spustí a zobrazí ji v popředí. Z této aplikace pak můžete spustit jinou aplikaci nebo jinou obrazovku stejné aplikace a tak dále. Všechny programy a obrazovky jsou ukládany do aplikačního zásobníku, o který se stará Activity Manager, jenž je součástí Android runtime popsaného v kapitole 2.4.3. V kterémkoliv okamžiku může uživatel stisknout tlačítko zpět a vrátit se tak k některé z předchozích obrazovek v zásobníku. Toto chování by se dalo přirovnat k chování internetového prohlížeče a používání tlačítek zpět a vpřed.

Každé uživatelské rozhraní je reprezentováno třídou Activity (obdoba formuláře ve Windows). Každá aktivita má svůj vlastní životní cyklus. Aplikace se skládá z jedné nebo více aktivit a linuxového procesu, který je jejich kontejner.

V OS Android může být aplikace naživu dokonce, i když už byl proces zabit. Tzn., že životní cyklus aktivity není pevně spojen s procesem. Procesy jsou pouze dočasné schránky pro aktivity

Během svého životního cyklu, může být aktivita v různých stavech. Jako programátor sice nemáte kontrolu nad stavy aktivity, o to se stará systém, ale systém Vás upozorní na změnu stavu a vy na ni můžete reagovat. Třída Activity obsahuje sadu metod pro jednotlivé stavy, které nastanou v momentě, kdy aktivita přejde do nového stavu, a Vy můžete tyto metody přepsat.

- *onCreate(Bundle)* nastane v momentě, kdy se aktivita poprvé spustí. Dá se využít například- použít třeba k první inicializaci proměnných apod. Jako parametr bere třídu Bundle, která buď obsahuje informace uložené v předchozím stavu, nebo může být null
- *onStart()* indikuje, že aktivita se chystá být zobrazena na displeji
- *onResume()* je vyvolána v momentě, kdy je aktivita připravena komunikovat s uživatelem

- *onPause()* nastane v momentě, kdy je aplikace připravena jít do pozadí. Obvykle proto, že byla spuštěna jiná aktivita. V této metodě byste si měli uložit všechny důležité věci jako třeba změněné záznamy databáze apod.
- *onStop()* metoda je vyvolána ve chvíli, kdy aktivita už není zobrazena uživateli a nebude po nějakou dobu potřeba. Musíte mít ale na paměti, že pokud systém nebude mít dostatek paměti, může místo volání onStop() aplikaci jednoduše ukončit
- *onRestart() p*okud je zavolána tato metoda znamená to, že aktivita se chystá být zobrazena, po tom co byla ve stavu stop
- *onDestroy()* je vyvolána v momentě, kdy se chystá ukončení aktivity. Stejně jako onStop, nemusí být vyvolána, pokud má systém nedostatek pamět.
- *onSaveInstanceState(Bundle)* Android zavolá tuto metodu, aby umožnil aktivitě uložit si své instanční proměnné, jako jsou pozice kurzoru v textovém poli atd. Obvykle, ale není třeba přepisovat tuto metodu, protože její implementace sama uloží stav všech prvků aktivity
- *onRestoreInstanceState(Bundle)* nastane, když je aktivita opět inicializována ze stavu, který předtím uložila metoda onSaveInstanceState(Bundle). Výchozí implementace obnoví stav uživatelského rozhraní

Aktivity, které neběží v pozadí mohou být zastaveny. Linuxový proces, který je hostí může být zabit v jakémkoliv momentě, aby se uvolnila paměť. Toto chování je poměrně běžné, proto je důležité aby s tímto chováním počítala Vaše aplikace již od začátku. V některých případech může být *onPause()* poslední metodou, která bude zavolána. Proto byste v této metodě měli uložit všechna důležitá data.

2.6 Vývoj pro Android

Android spojuje všudy přítomnost mobilních telefonů, a to otevřený software open source s širokou podporou ze strany Google a dalších Open Handset Alliance členů jako je Motorola, HTC, Verizon a AT & T. Naštěstí pro vývojáře, začít vývoj pro operační systém android není až tak složité, jak by se zdálo. Není ani potřeba mít k dispozici zařízení s operačním systémem android. Stačí jen počítač, kde si můžete nainstalovat Android SDK a emulátor telefonu.

2.6.1 Instalace SDK

Software Development Kit (SDK), je k dispozici pro Windows, Linux, i Mac OS X. Aplikace, které vytvoříte, samozřejmě můžete nainstalovat na jakékoli zařízení se systémem Android.

2.6.1.1 Java

Než začnete s instalací Android SDK musíte mít nejprve nainstalovanou kopii Java (verze 5 nebo verze 6), vyžadují ji všechny vývojové nástroje Android. Stejně tak jí budou využívat všechny aplikace, které napíšete. Nestačí mít nainstalované pouze Java Runtime Environment, je nutné nainstalovat si celý balíček Java Development Kit (JDK). Pokud si potřebujete ověřit jakou verzi Java máte nainstalovanou, stačí napsat do příkazového řádku příkaz „java –version“

```
C:\> java -version

java version "1.6.0_14"

Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
```

Kód 1 Ověření verze Java

2.6.1.2 Eclipse

Samozřejmě máte možnost zvolit si vývojové prostředí dle Vašeho uvážení. Doporučil bych ale Eclipse, které je zdarma a má poměrně širokou podporu ze strany Google. Minimální verze Eclipse, ve které lze vyvíjet Android aplikace, je 3.3.1, ale měli byste vždy používat aktuální verzi. Pro vývoj Android aplikací se doporučuje používat verzi Eclipse Classic. Po stažení stačí soubor rozbalit do libovolné složky a můžete začít používat Eclipse.

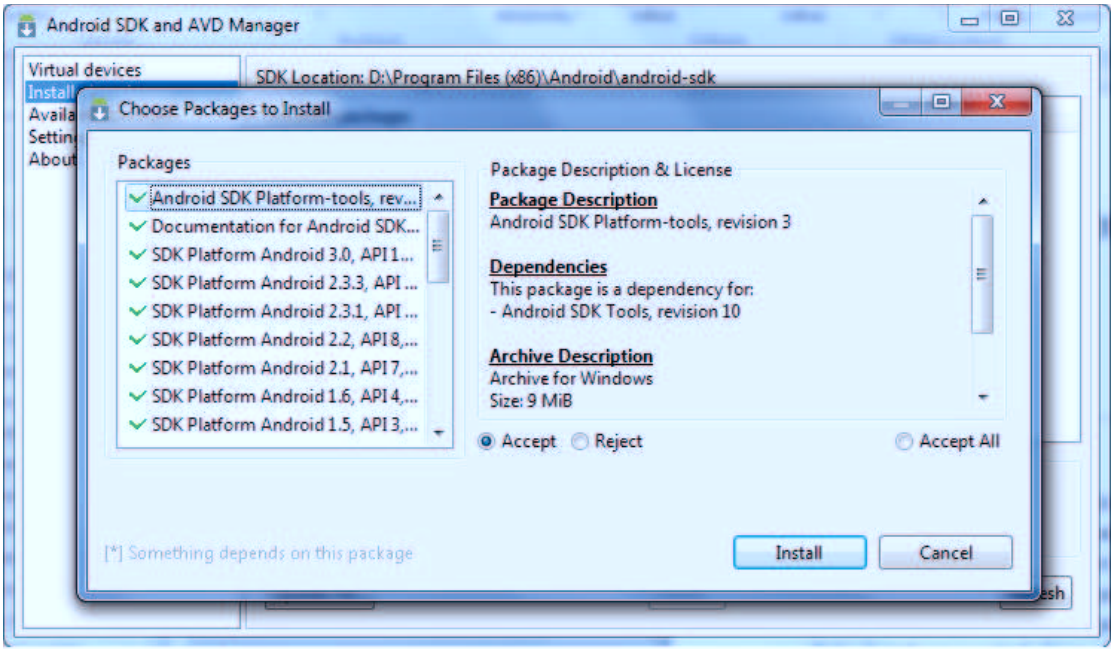
Pokud nechcete použít Eclipse, existuje i podpora pro jiný IDE, jako jsou NetBeans a JetBrains IDEA, ale jejich podpora není tak dobrá jako pro Eclipse. Dokonce se můžete vzdát IDE zcela a používat jen příkazový řádek.

2.6.1.3 Android SDK Starter Package

Od verze Android 2.0 byl Android SDK rozdělen do dvou částí: SDK Starter Kit a SDK komponenty.

SDK Starter Kit je balíček, který se dá stáhnout z webu vývojářů Android. Po jeho stažení a nainstalovaní se spustí SDK Setup.exe. Instalátor nyní zobrazí seznam dostupných komponent včetně dokumentace, platforem, knihoven a USB ovladačů. Můžete si zvolit, které části SDK chcete nainstalovat. Například pro vývoj Vám bude stačit SDK verze 2.2 a nemusíte stahovat jiné verze SDK. Podle toho, které SDK si zvolíte, můžete pak vytvářet aplikace pro jednotlivé verze Android. Stažené komponenty se dají kdykoliv změnit nebo aktualizovat opětovným spuštěním SDK Setup.exe. Instalaci spustíte tlačítkem „install“. Stažené soubory budou uloženy do složky, ve které je SDK Setup.exe. Stažení

bude trvat nějakou chvíli. Kompletní instalace zabírá něco přes 2GB na disku.

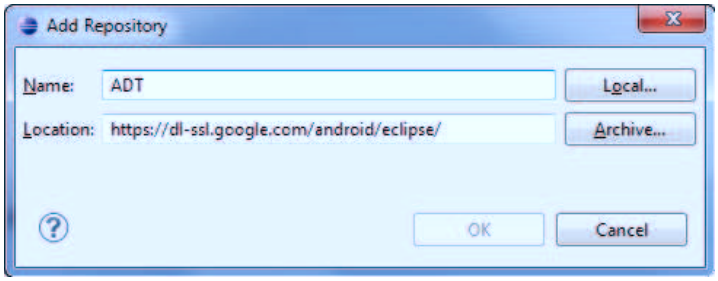


Obrázek 8 Instalace Android SDK

2.6.1.4 Android Development Toolkit (ADT)

ADT plugin vytvořil Google pro Eclipse, aby usnadnil vývoj aplikací pro operační systém Android. Za předpokladu, že máte nainstalovaný výše uvedený software, je instalace rozšíření opravdu jednoduchá. Stačí jen postupovat podle následujících kroků:

1. Spusťte Eclipse.
2. V nabídce „Help“ vyberte „Install New Software“.
3. V zobrazeném dialogu klikněte na tlačítko „Add“ tím přidáte nový repositář.
4. Jako název do pole „Name“ zadejte například „ADT“, do pole „Location“ zadejte adresu „https://dl-ssl.google.com/android/eclipse/“ a potvrďte tlačítkem „Ok“.



Obrázek 9 Dialog repositáře Eclipse pro stažení ADT

5. Zatrhněte všechny zaškrtávací pole a klikněte na tlačítko „Next“.
6. Po stažení rozšíření je třeba restartovat Eclipse. Po restartu už můžete začít vyvíjet aplikace pro Android.

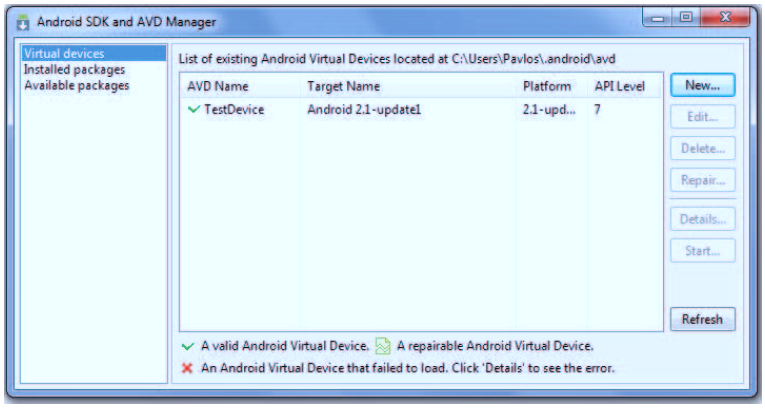
2.6.2 Emulátor a správce virtuálních zařízení

Jak už bylo zmíněno, ke spuštění nebo testování aplikací nemusíte mít k dispozici zařízení s Androidem. Aby byl vývoj jednoduchý opravdu pro každého Google přidal do SDK emulátor, který využívá stažených balíčků z SDK. Díky tomu můžete vyvíjet aplikace pro nejnovější verzi Androidu a máte jistotu, že bude plně funkční na cílovém systému.

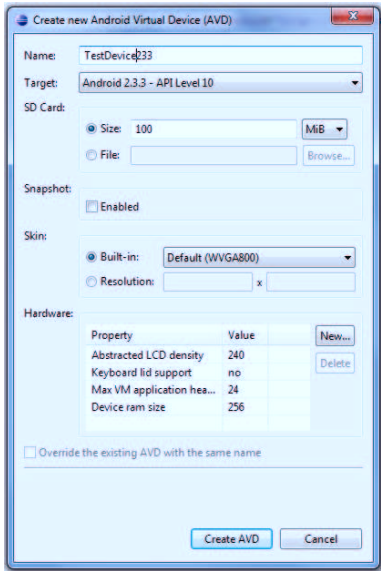
2.6.2.1 Android Virtual Device (AVD)

Abyste mohli spouštět aplikace na emulátoru musíte nejprve vytvořit virtuální zařízení s Androidem. Zařízení se vytváří v manažeru virtuálních zařízení, který se jmenuje AVD Manager (Obrázek 10). Lze ho spustit přímo z Eclipse přes Window > Android SDK > AVD Manager nebo přes instalační program „SDK Setup.exe“ v záložce „Virtual Devices“. Můžete si vytvořit i více zařízení s různými cílovými verzemi androidu a ověřit si funkčnost aplikace na každé z nich.

Vytvoření virtuální zařízení je velice jednoduché. V AVD Manager-u stačí kliknout na tlačítko „New“ a v dialogu (Obrázek 11) vyplnit název zařízení, cílovou verzi. Můžete také nastavit specifické hardwarové závislosti, jako je velikost SD karty, rozlišení a typ displeje. A také další hardwarové specifikace jako jsou například, podpora GPS, akcelerometr, trackball, kamera proximity senzor nebo telefonní modul.



Obrázek 10 AVD Manager



Obrázek 11 Nové AVD

Při spuštění ladění aplikace z prostředí Eclipse se při prvním spuštění otevře dialogové okno, ve kterém si zvolíte virtuální zařízení, na kterém chcete testovat aplikaci. Běh emulátoru je při prvním spuštění poněkud pomalý, než naběhne celý systém. Avšak při opětovném spouštění aplikace zůstává emulátor zapnutý a nemusí startovat od začátku. Pouze nainstaluje a spustí novou verzi aplikace.

2.6.3 Spuštění na reálném zařízení

Spuštění programu Android na fyzickém zařízení jako Nexus One nebo jiných je v průběhu vývoje téměř identické s vývojem na emulátoru. Musíte jen na samotném zařízení povolit USB ladění (Nastavení > Aplikace > Vývoj > USB Ladění). Je třeba mít nainstalované ovladače Android USB.

Jestli máte spuštěné okno emlulátoru, tak jej zavřete. Pokud je telefon správně připojený k PC pomocí datového kabelu a má povolené USB ladění, bude Eclipse pro běh

aplikací používat připojené zařízení místo emulátoru. Výhoda reálného zařízení je zejména v rychlosti ladění.

3 Open XML

Mezi mnoha novými technologiemi implementovanými do Microsoft Office 2007 je jedna, kterou byste neměli přehlédnout. Je to nový značkovací jazyk Open XML pro dokumenty sešity a prezentace. Jeho hlavní výhodou je zjednodušení vývoje dokumentů a velikosti výsledných souborů oproti předcházejícím binárním souborům. Open XML poskytuje otevřené a standardizované prostředí, které staví na mnoha existujících standardech, jako jsou XML, ZIP nebo XML Schema. Tyto technologie jsou široce používané ve všech dnešních operačních systémech. Dokument již není pouze černá skříňka obsahující formátovaná data. Místo toho se z dokumentu stala data, které lze snadno integrovat do podnikových procesů nebo dále zpracovávat.

3.1 Historie

OOXML však nebyl prvním pokusem pro specifikaci dokumentů kancelářských balíků. Jeho předchůdcem byly „Microsoft Office XML Formats“, které jsou také založeny na XML a XSD souborech. Microsoft na této specifikaci začal pracovat a v roce 2000 přišel s podporou XML pro tabulkový procesor Excel. Specifikace byla následně zahrnuta do kancelářského balíku Microsoft Office XP, zatím ale jen s podporu tabulek Excel. To se ale brzy změnilo. Hned v roce 2002 přibyla i podpora pro dokumenty Word, která byla zahrnuta do Microsoft Office 2003. Zatím se však jednalo o samostatné XML soubory, které nebyly zabaleny. Nevýhodou pak byly o mnoho větší sobory v porovnání s binárním souborem. Microsoft Office XML Formats se pak staly předlohou při tvorbě specifikace, který z této specifikace vychází.

Formáty Office XML:

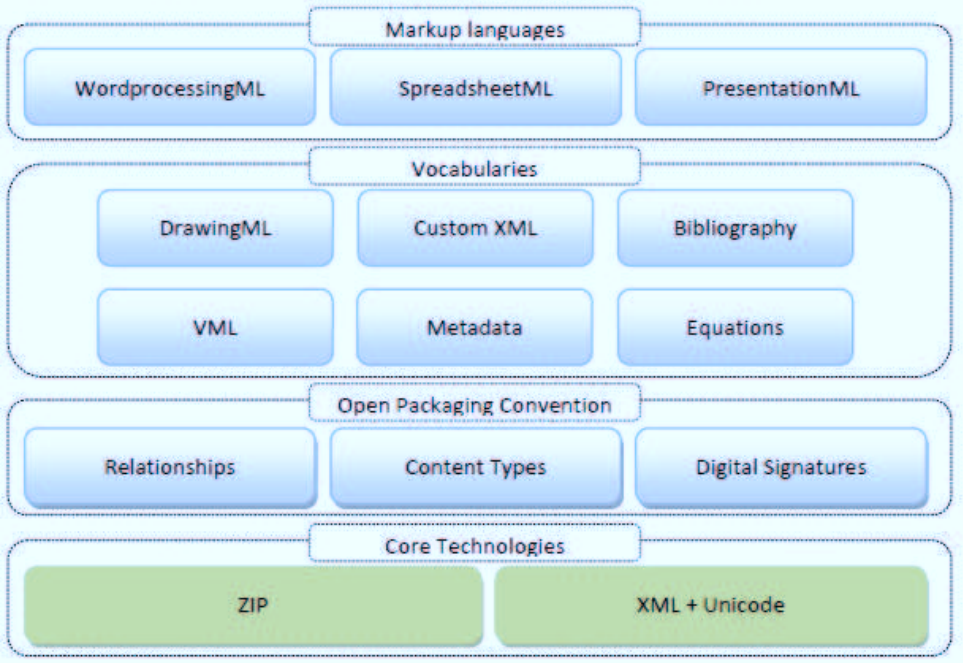
- Microsoft Office Word 2003 XML Format — WordProcessingML or WordML (.XML)
- Microsoft Office Excel 2002 and Excel 2003 XML Format — SpreadsheetML (.XML)
- Microsoft Office Visio 2003 XML Format — DataDiagramingML (.VDX, .VSX, .VTX)
- Microsoft Office InfoPath 2003 XML Format — XML FormTemplate (.XSN) (Compressed XML templates in a Cabinet file)

3.2 Standard Open XML

Dřívější verze Microsoft Office používali binární formát souborů. Teprve až Microsoft Office 2007 představil standart značkovacího jazyka Open XML. Nové formáty office jsou založené na XML a jsou dnes široce rozšířené v mnoha softwarových platformách nebo operačních systémech. Open XML obsahuje 3 základní značkovací jazyky. Je zde WordProcessingML pro dokumenty, SpreadsheetML pro sešity a PresentationML pro prezentace. Je zde také mnoho dalších podpůrných značkovacích jazyků jako třeba DrawingML pro kreslení grafiky, grafů, tabulek a diagramů.

Dokument Open XML je uložen jako ZIP kontejner obsahující mnoho částí. Jednotlivé části jsou pak uvnitř tohoto kontejneru a lze je procházet jako každý jiný ZIP archív. Struktura kontejneru je standar-

dizovaná pod názvem Open Packaging Conventions. Obrázek 12 ukazuje přehled vrstev architektury specifikace.



Obrázek 12 Architektura Open XML

3.3 Části specifikace

Open Packaging Conventions (OPC)

Definuje vnitřní strukturu ZIP archívu, v kterém je uložen dokument. Udržuje informace o vztazích mezi jednotlivými soubory dokumentu. Informace o vazbách a vztazích mezi soubory jsou uloženy v souboru s příponou *.rels*. Příkladem takovýchto vazeb může být například vztah mezi sešitem a jeho listy pro tabulkový kalkulátor.

WordProcessingML

Specifikace XML souborů pro textové dokumenty. Základní struktura vychází se schématu body – paragraph – run . Tzn., že tělo dokumentu (body) je členěno do odstavců(*paragraph*) a odstavce obsahují pak formátovací informace (*run*). Element run obsahuje informace o formátování podřízeného elementu <*w:t*>, který obsahuje samotný text.

SpreadsheetML

Specifikace pro soubory tabulkového kalkulátoru. Základní struktura vychází ze schématu workbook-sheet. Už z názvu je jasné, že kořenovým elementem je workbook, který obsahuje nastavení sešitu. Jako je např.: formát stránky, okraje, zvolené zobrazení, informace o autorovi a mnoho dalších. Workbook obsahuje elementy sheet, které reprezentují jednotlivé listy sešitu. Sešit i listy jsou umístě-

ny v samostatných souborech. Typicky workbook.xml a sheetN.xml, kde N je číslo sešitu. Pro ukládání textové informace ve SpreadSheetML existují dva způsoby. Textovou informaci lze uložit přímo do buňky (inlinestring) nebo jako sdílený řetězec, který je uložen v samostatném souboru obsahující jiné sdílené řetězce (sharedstring), a je společný pro všechny listy v sešitu. Místo textu je pak v buňce uložen pouze odkaz na sdílený string.

PresentationML

Popisuje definici XML pro prezentace. Obsah dokumentu se dělí na snímky (*slide*), jejich obsah a animace. Každý snímek je podobně jako ve SpreadSheetML uložen v samostatném XML souboru.

DrawingML

Popisuje XML pro grafické elementy dokumentů. DrawingML popisuje vektorovou grafiku, která se používá v rámci jiných ML specifikací jako: WordProcessingML, SpreadsheetML a PresentationML.

Ostatní „ML“

Kompletní specifikace Open Xml obsahuje 6546 stran, proto jsou uvedeny jen ty nejdůležitější části. Specifikace obsahuje ještě mnohé další definice. Např. sada pro zápis matematických vzorců (Office MathML) nebo sady pro různé specifické úkoly

Custom XML

Definuje způsob jak vkládat uživatelská XML do dokumentu a propojovat je s obsahem.

3.4 WordProcessingML

3.4.1 Digitální dokumenty

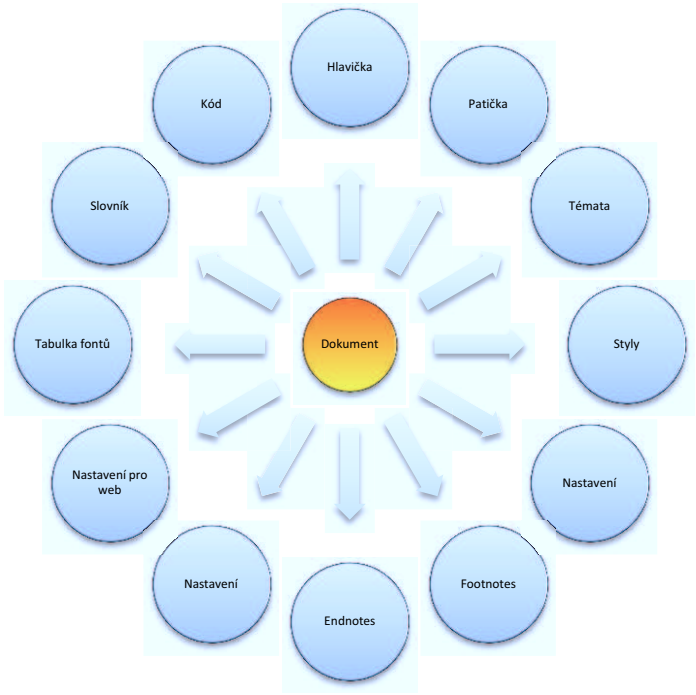
Dokumenty se používaly daleko dříve, než vznikl nápad vytvořit elektronické dokumenty. Tyto dokumenty se vytvářely pomocí různých zastaralých nástrojů, jako třeba psacích strojů a nedosahovali ani zdaleka takových možností jako dnešní digitální dokumenty.

Využití dokumentů také prošlo velkou změnou. Použití elektronických dokumentů přináší oproti papírovým mnohé výhody, jako jsou například digitální podpis, reference na jiné dokumenty, grafy, odkazy a mnohé jiné. Tyto vlastnosti umožňují širokou použitelnost pro zpracování dat v dokumentech, zejména v podnikových procesech.

3.4.2 Základní struktura

Pokud vezmeme dokument WordProcessingML a v průzkumníkovi změníme jeho příponu z *docx* na *zip*. Objevíme uvnitř archívu spoustu různých souborů, zejména v rozsáhlejších dokumentech. WordProcessingML dokument se dělí na různé části a každá část je popsána v samostatném souboru uvnitř archivu. Kromě souborů, které ukládají data pro dokument jsou zde další soubory pro ukládání nastavení, tabulek fontů nebo stylů. Obrázek 13 popisuje adresářovou strukturu WordProcessingML dokumentu.

V kořenovém adresáři archívu se nachází soubor *[Content_Types].xml*. Tento soubor obsahuje slovník, který popisuje typ obsahu všech ostatních souborů v archivu. Zde je velký rozdíl oproti binárním souborům, které obsahovaly všechna data dohromady v jednom souboru. Místo toho je obsah rozdělen do více částí a u každé části je definováno jaká data v ní může uživatel očekávat.



Obrázek 13 Struktura WordProcessingML dokumentu

Minimální dokument WordProcessingML musí obsahovat alespoň tři základní části. Je potřeba soubor *document.xml*, který definuje hlavní tělo dokumentu. Tento soubor potřebuje uložit svůj typ v souboru *[Content_Types].xml*. Hlavní část dokumentu musí být umístitelná pomocí odkazu, který je popsán v třetí části, která se zabývá odkazováním na jiné části dokumentu.

K vytvoření základního dokumentu stačí pouze vytvořit adresář. Uvnitř tohoto adresáře vytvoříme složku *_rels*. Ve složce *_rels* je uložena právě ta část, která se zabývá odkazováním mezi částmi dokumentu. Další dvě základní části dokumentu jsou pak uloženy v souborech *[Content_Types].xml* a *document.xml*.

Prvním krokem k vytvoření dokumentu je právě soubor *document.xml*. Pokud bychom parsovali dokument, začíná se právě tímto souborem. Každý ze tří značkovacích jazyků Open XML má svou startovní část. V případě WordProcessingML je v tomto souboru uloženo právě hlavní tělo dokumentu. Základní částí je element *document*, uvnitř kterého pak můžete používat různé stavební bloky jako jsou tabulky nebo odstavce. Většina elementů uvnitř tohoto souboru používá stejný jmenný prostor http://schemas.openxmlformats.org/wordprocessingml/2006/main.

```
<?xml version="1.0" encoding="UTF-16" standalone="yes" ?>
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
</w:document>
```

Kód 2 WordProcessingML - Minimální soubor document.xml

Nyní když je nadefinována startovní část, je třeba nastavit typ obsahu, aby Open XML parser mohl najít informaci jaký typ značek je uložen uvnitř části dokumentu. Typ obsahu je veden jako seznam uvnitř balíčku. Tento soubor se musí nacházet v kořenovém adresáři balíčku a být pojmenovaný *[Con-*

tent_Types].xml. Uvnitř elementu *Types*, který definuje použité typy, najdete dva elementy a to *default*, který určuje typ obsahu pro určitou příponu. Například pro xml je to „application/xml“. Element *override* upravuje typ souboru document.xml, jelikož *document.xml* obsahuje hlavní část dokumentu, musí mít jiný typ než je „application/xml“, proto je přepsán typ elementem *override.*

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
  <Default Extension="rels" ContentType="application/vnd.openxmlformats-
package.relationships+xml" />
  <Default Extension="xml" ContentType="application/xml" />
  <Override PartName="/document.xml" ContentType="application/vnd.openxmlformats-
officedocument.wordprocessingml.document.mainxml" />
</Types>
```

Kód 3 WordProcessingML - Minimální soubor [Content_Types].xml

Ačkoliv balíček obsahuje mnoho částí, které jsou spolu propojené, je zde jedna část, která je speciální tím, že obsahuje seznam odkazů na startovací soubor. Tyto vazby nebo též relace jsou uloženy ve speciálním souboru, který končí příponou *.rels*. Ty jsou uloženy uvnitř složky *_rels,* která se nachází v kořenovém adresáři balíčku. Soubor *.rels* obsahuje jedinou relaci popisující hlavní část dokumentu *document.xml*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
        <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/
2006/relationships/officeDocument" Target="document.xml" />
</Relationships>
```

Kód 4 WordProcessingML - Minimální soubor .rels

To jsou tři základní částí každého WordProcessingML dokumentu. Balíček obsahuje samozřejmě daleko více specifických částí, jejichž vysvětlení je nad rámec této práce. Dokumentace těchto částí má něco přes 6500 stránek.

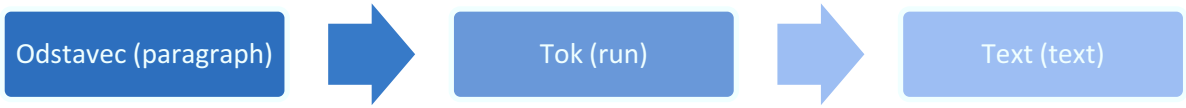
3.4.3 Organizace a formátování obsahu dokumentu

Asi nejčastější věcí, kterou obsahuje každý dokument, je nějaký text. Většina textů je uložena v hlavní části dokumentu *document.xml*. Další výskyt může být buďto v hlavičce nebo patičce, ale ty jsou uloženy v samostatných částech. Kořenový element je *document*, který obsahuje jeden element *body*, ve kterém je definováno tělo dokumentu. Element *body* obsahuje stavební bloky ze kterých je sestaven celý dokument. Mohou být například, odstavce, tabulky a obrázky. V elementu *body* může být uložen obsah, který je buď blokový nebo řádkový. Blokovým obsahem jsou například odstavce nebo tabulky. Blokové elementy obsahují řádkové elementy. Obvyklým typem řádkových elementů jsou formátovaný text nebo obrázek.

3.4.4 Odstavce

Asi nejzákladnější věc, kterou bude dokument obsahovat je text v odstavci. Text je většinou uložen v hlavní části dokumentu. Další části, kde se může text objevit jsou například hlavičky nebo patičky, ale ty jsou uloženy v samostatných souborech.

Odstavec je rozdělen do několika toků (anglicky runs). Element *run* je nejnižší element v hierarchii, který může obsahovat formátovaný text. Element *run* obsahuje několik textových elementů *text*. Element *text* definuje tisknutelný text. Existuje ovšem i element pro netisknutelné znaky, jako jsou konce řádků nebo zalomení stránek. Odstavec je základním blokem v dokumentu, proto je nejlepší volbou pro vhodné nastavení odsazení a tabulátorů.



V následující ukázce je popsána nejjednodušší verze souboru *document.xml*, která obsahuje pouze jeden odstavec s textem „Hello World!“ Text uvnitř elementu *<w:t>* může být klidně rozdělen na více částí, ale zobrazený text bude stejný.

```
<?xml version="1.0" encoding="UTF-16" standalone="yes" ?>
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
        <w:body>
                <w:p>
                        <w:r>
                                <w:t>Hello World!</w:t>
                        </w:r>
                </w:p>
        </w:body>
</w:document>
```

Kód 5 WordProcessingML - ukázka Hello World

3.4.5 Formátování textu

Další důležitou vlastností při tvorbě dokumentu je formátování textu. Abychom mohli zformátovat část textu, nabízí se hned několik možností. Nejjednodušší množností je aplikovat formátování přímo v odstavci v element toku textu (*run*). Pokud se styl bude často opakovat, nabízí se jako lepší možnost vytvořit styl, který se dá použít vícekrát.

Běžný text bez použití stylů se dá formátovat buď na úrovni odstavce (*paragraph*) nebo toku textu (*run*). Obě možnosti nabízejí spoustu různých nastavení, které je nejlépe vidět přímo v nastavení odstavce v aplikaci Microsoft Word. Formátování odstavce obsahuje v základu nastavení, která se týkají celého odstavce, jako jsou mezery před a za, odsazení nebo řádkování. Nastavení toku umožňuje nastavení věci okolo znaků samotných, jako je font, tučnost a jiné.

Formátování textu (element run)

Element, ve kterém je uloženo nastavení formátování toku textu – velikost a název fontu, tučnost atd. se nazývá *run-properties* a značí se *rPr*. Každý element run-properties musí mít nastaven název a velikost fontu. Volitelným nastavení jsou třeba barva písma nebo řádkování, tučnost nebo kurzíva.

V následující ukázce je zobrazen text „ukázka textu” který má nastavený font Cambria velikosti 26 bodů, nastavenou barvu, je tučný, kurzívní a podtržený. Tučný text se nastaví pomocí elementu

<w:b />, stejně tak kurzívní elementem *<w:i />*. Nastavení podtržení je element *<w:u />*, kterému se přidává atribut *val*, který udává typ podtržení. Například *single* nebo *double*. Element *<w:sz />* nastavuje velikost písma. Zde je zajímavé, že hodnota je udávána v takzvaných halfpoint, což jsou vlastně dvojnásobky klasických pointů. Tedy pokud chceme mít text velikost 26pt, hodnota atributu *val* v elementu *sz* bude 52. Zbývá *<w:color />*, který nastaví barvu textu. Hodnota je klasicky v hexa kódu, zastupují ji tři barevné složky. Červená, zelená a modrá ve tvaru RRGGBB, kde maximální hodnota každé složky je FF v šestnáctkové soustavě. Poslední zajímavou částí je element *rFonts* který nastavuje font. Pokud bychom chtěli mít různé části textu s různým formátováním, musíme je rozdělit na více *run* elementů a každý z nich bude obsahovat nastavení pro konkrétní část textu a blok *<w:t />* , který bude obsahovat samostatnou část textu.

```
<w:r>
        <w:rPr>
                <w:rFonts w:ascii="Cambria" />
                <w:b />
                <w:i />
                <w:u w:val="single" />
                <w:color w:val="548DD4" />
                <w:sz w:val="52" />
        </w:rPr>
        <w:t>Ukázka textu</w:t>
</w:r>
```

Kód 6 WordProcessingML - formátování toku textu

Formátování odstavce (element paragraph)

Nastavení odstavce jsou uložena uvnitř elementu *paragraph-properties*, zkráceně *pPr*. Nastavení se ukládá stejným způsobem jako v *rPr*. Odstavec je považován za blokový element, proto je obvykle široký tak, jak jenom stránka umožňuje.

Uvnitř elementu *pBdr*, který nastavuje ohraničení odstavce, můžeme podobně jako v HTML definovat ohraničení okolo odstavce. Nastavit můžeme i typ a šířku ohraničení. Nastavení mezery před a za dokumentem se nastavuje v elementu *spacing*. Atribut *before* a *after* udávají velikost odsazení. Jednotka je twips, což je 1/20 pointu. Tedy 120 twips odpovídá 6pt. Odsazení první řádky v odstavci se definuje v elementu *ind* a atributu *firstLine*. Jednotky jsou opět twips. Pokud je text zarovnán v odstavci vpravo, na střed nebo do bloku, je uveden element *jc* s hodnotou určující zarovnání.

```
<w:pPr>
        <w:pBdr>
          <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto" />
        </w:pBdr>
        <w:spacing w:before="120" w:after="240" />
        <w:ind w:firstLine="709" />
        <w:jc w:val="right" />
</w:pPr>
```

Kód 7 WordProcessingML - formátování odstavce

3.4.6 Zalomení

Ve WordProcessingML existují dva různé typy zalomení. Jedno zalomení se používá pro zalomení řádků nebo konec stránky, který zalomí aktuální stránku. Obojí zalomení se používají pouze v elementu toku textu (*run*). Zalomení řádku se označuje elementem <*w:cr* /> a zalomení stránky elementem <*w:br* />. Element *br* se dá také použít k zalomení sloupce s tím rozdílem, že atribut *type* je typu *column*.

```
<w:r>
        <w:cr />
</w:r>
<w:r>
        <w:br w:type="page" />
</w:r>
```

Kód 8 WordProcessingML - Ukázka zalomení

3.4.7 Tabulky

Hned po odstavcích jsou tabulky druhé nejčastěji používané stavební bloky dokumentu. Tabulky jsou tvořeny podobně jako v HTML. Tedy jsou tvořeny z řádků a buněk. Stejně jako odstavec je reprezentován elementem *p* má i tabulka svůj element *tbl*. Řádky jsou definovaný *tr* (table-row) a buňky *tc* (table-cell). Uvnitř buňky už se pak může opět nacházet třeba odstavec nebo obrázek, popřípadě i další tabulka. Kód 9 WordProcessingML - Tabulka 2x3 ukazuje základní definici tabulky o 2 řádcích a 3 sloupcích. Místo … by pak byly elementy reprezentující text v dané buňce.

```
<w:tbl>
        <w:tblGrid />
        <w:tr>
                <w:tc>...</w:tc>
                <w:tc>...</w:tc>
                <w:tc>...</w:tc>
        </w:tr>
        <w:tr>
                <w:tc>...</w:tc>
                <w:tc>...</w:tc>
                <w:tc>...</w:tc>
        </w:tr>
</w:tbl>
```

Kód 9 WordProcessingML - Tabulka 2x3

Důležitým elementem je *tblGrid*, který obsahuje nastavení mřížky. Hlavní úlohou tohoto elementu není určovat počet sloupců, ty se určí podle počtu buněk v řádku. Ale ukládají se zde výchozí nebo preferované hodnoty pro šířku sloupce. Ale i každá buňka musí mít nastavenou svou šířku. V *tblGrid* je pouze preferovaná hodnota.

Ve výchozím nastavení se tabulka automaticky rozloží rovnoměrně, tak aby vyplňovala celou použitelnou šířku stránky. Pokud bychom potřebovali nastavit šířku tabulky, je potřeba přidat element *tblPr*. Viz. Kód 10 WordProcessingML - šířky sloupců v tabulce. Jak už bylo řečeno, každá buňka má také

nastavenou svou šířku. Šířka buňky se nastavuje v elementu *tcPr* (table-cell properties). Jednotkou šířky jsou opět twips spolu s atributem určujícím typ jednotky na *dxa*.

```
<w:tbl>
        <w:tblPr>
                <w:tblW w:w="0" w:type="auto" />
        </w:tblPr>
        <w:tblGrid>
                <w:gridCol w:w="1134" />
                <w:gridCol w:w="1134" />
                <w:gridCol w:w="1134" />
        </w:tblGrid>
        <w:tr>
                <w:trPr>
                        <w:trHeight w:val="454" />
                </w:trPr>
                <w:tc>
                        <w:tcPr>
                                <w:tcW w:w="1134" w:type="dxa" />
                        </w:tcPr>
                        <w:p>
                                <w:r><w:t>1</w:t> </w:r>
                        </w:p>
                </w:tc>
          </w:tr>
</w:tbl>
```

Kód 10 WordProcessingML - šířky sloupců v tabulce

V *tcPr* se nastavuje také sloučení buněk. A to pokud je buňka sloučena vertikálně obsahuje element *<w:gridSpan w:val="?" />*, kde *?* určuje počet buněk, které slučuje. Pokud je sloučena třeba přes tři buňky, bude to místo otazníku trojka. Pokud jsou buňky sloučeny, je v dokumentu uvedena stejně jako v HTML jenom ta první buňka s nastavením sloučení. Trochu jiný zápis se používá pokud se buňka slučuje vertikálně tj. přes řádky. Pokud jsou buňky sloučeny horizontálně uvádí se jen první buňka, pokud jsou ale sloučeny vertikálně, musí být definována každá buňka. Počátek vertikálního sloučení se pak označí elementem *<w:vMerge w:val="restart" />*, který je potomkem elementu *tcPr*. Další buňky, které jsou sloučené pod touto buňkou obsahují prázdný element *vMerge* bez hodnoty *restart*.

Když otevřete dokument, který obsahuje tabulku, poznáte, že jde o tabulku podle ohraničení buněk. Toto ohraničení není automatické. Musíte ho nastavit buďto v nastavení vlastností tabulky nebo buňky (*tblPr* nebo *tcPr*). Open Xml umožňuje definovat osm typů ohraničení. Jsou zde základní ohraničení jako vlevo, vpravo, nahoře a dole. Mimo to lze nastavit horizonální linky, vertikální linky a dvě diagonální ohraničení. Pro každé ohraničení je třeba také uvést typ ohraničení. Například jednoduché nebo dvojité. Pokud se bude nastavovat ohraničení tabulky jako celku musí se přidat element *tblBorders* jako potomek *tblPr*. Naopak pokud je potřeba nastavit ohraničení buňce, je nutné vytvořit element *tcBorders* uvnitř *tcPr*. Šířka ohraničení je uvedena jako 1/8pt. Platný rozsah je od 2 do 96. Kód 11 WordProcessingML - Nastavení ohraničení tabulky a buňky ukazuje zdrojový kod pro tabulku, která nastavuje horní a spodní okraj pro celý obsah tabulky a element *tcPr*, který nastavuje dodatečné ohraničení buňky vlevo a vpravo. Pokud bychom chtěli nastavit ohraničení pro celý řádek, tak to není možné. Každá buňka musí obsahovat své nastavení ohraničení v elementu *tcBorders*.

```
 <w:tblPr>
        <w:tblBorders>
          <w:top w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
            <w:bottom w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
        </w:tblBorders>
</w:tblPr>

 <w:tcPr>
        <w:tcBorders>
            <w:left w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
            <w:right w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
        </w:tcBorders>
</w:tcPr>
```

Kód 11 WordProcessingML - Nastavení ohraničení tabulky a buňky

3.4.8 Styly

Pokud je v dokumentu obsaženo hodně formátování, je vhodné vytvořit tzv. styly. Jsou vlastně kolekce nastavení pro formátování textu. Doposud se formátování aplikovalo přímo na elementy odstavce, tabulky nebo toku textu. Styl umožní toto nastavení definovat pouze jednou a po té opakovaně použít stejnou sadu nastavení. Další výhodou je, že pokud použijete styly, můžete snadno měnit vzhled všech prvků, které mají daný styl přiřazen.

Styl je sada hodnot pro formátování, která může být samostatně aplikována na odstavce (*paragraphs*), toky textu (*runs*) nebo tabulky (*tables*). Dle specifikace WordProcessingML jsou styly uloženy v samostatném souboru *styles.xml* v libovolné složce, avšak soubor *[Content_Types].xml*

musí obsahovat relativní cestu k souboru a typ obsahu. Styly používají speciální typ obsahu *application/vnd.openxmlformats-officedocument.wordprocessingml.styles+xml*

```
<Override PartName="/word/styles.xml"
ContentType="application/vnd.openxmlformats-officedocument.
wordprocessingml.styles+xml" />
```

Kód 12 WordProcessingML - definice souboru, kde se nacházejí styly

Soubor *styles.xml* může obsahovat tři různé části. Výchozí nastavení dokumentu (*document defaults*), skryté styly (*latent-styles*) a obyčejné styly (*styles*). Obyčejné styly obsahují formátování, které je aktuálně použité v dokumentu. Skryté styly nejsou použité nebo viditelné v dokumentu, obsahují a nabízí pouze uložené výchozí styly, můžou být ignorovány. Další důležitou částí jsou výchozí nastavení dokumentu. Tyto vlastnosti jsou pak výchozí pro všechny prvky, které nemají přiřazený styl.

Každý styl může definovat nastavení pro tok textu (*run*), odstavec (*paragraph*) a tabulky (tables). Nastavení těchto vlastností je stejné jako v předchozích ukázkách formátování toku textu, formátování odstavce a formátování tabulek s tím rozdílem, že elementy *pPr, rPr* a *tblPr* jsou potomky elementu *style*. Element, který chce využít styl, pak musí obsahovat nastavení svých vlastností a mít uveden vnořený element *pStyles* (viz. Kód 13 WordProcessingML - Použití stylu v odstavci). Tedy pokud chci aby odstavec použil nějaký existující styl, musí obsahovat element *pPr* a ten vnořený element *<w:pStyle w:val="název"/>*, kde název je název stylu, který má být použít.

```
<w:p>
        <w:pPr>
                <w:pStyle w:val="Title" />
        </w:pPr>
        <w:r>
                <w:t>Stephen Jiang</w:t>
        </w:r>
</w:p>
```

Kód 13 WordProcessingML - Použití stylu v odstavci

Soubor *styles.xml* obsahuje také výchozí nastavení fontu. Lze uložit dva typy výchozích hodnot. A to pro odstavec a pro znaky (tok textu). Tato nastavení používají stejné elementy jako přímé nastavení formátu. Je zde *pPr* pro nastavení odstavce, *rPr* pro nastavení znaků. Tyto elementy jsou podřízeny speciálním elementům *pPrDefault* a *rPrDefault*. Kód 14 WordProcessingML - Nastavení výchozích hodnot odstavce a toku textu ukazuje, jakým způsobem jsou organizovány výchozí nastavení pro od-

stavec a tok textu.

```
<w:styles xmlns:w="http://.../wordprocessingml/2006/main">
        <w:docDefaults>
                <w:rPrDefault>
                        <w:rPr>
                                <w:rFonts w:ascii="Calibri" />
                                <w:sz w:val="22" />
                        </w:rPr>
                </w:rPrDefault>
                <w:pPrDefault>
                        <w:pPr>
                                <w:spacing w:after="120" />
                        </w:pPr>
                </w:pPrDefault>
          </w:docDefaults>
</w:styles>
```

Kód 14 WordProcessingML - Nastavení výchozích hodnot odstavce a toku textu

Pokud jste v Microsoft Word někdy vytvářeli vlastní styl. Určitě jste si všimli, že je pět možností, které určují typ stylu. Jsou to odstavec, znak, propojený (odstavec + znak), tabulka a seznam. Styl může být navíc založen na jiném existujícím stylu. Pouze ho rozšíří o nové nastavení nebo přepíše konkrétní nastavení.

Kód 15 ukazuje styl založený na odstavci. Nastavení formátování odstavce a toku textu bude platné pro celý odstavec. *StyleId* určuje název stylu, pro odkazování se v dokumentu. *Name* definuje uživatelsky přívětivé jméno stylu. Tento styl může také obsahovat element *next*, který slouží k odkázání na styl, který bude použit v dalším odstavci. Pokud styl vychází z jiného může obsahovat element *basedOn* s atributem určujícím název stylu, na kterém je založen. Příklad by dále mohl obsahovat nastavení jako v předchozím příkladu Kód 14 elementů *rPr* a *pPr*.

```
<w:style w:type="paragraph" w:styleId="MujOdstavec">
        <w:name w:val="MujOdstavec" />
        <w:next w:val="Normal" />
        <w:basedOn w:val="Normal" />
        <w:qFormat />
</w:style>
```

Kód 15 WordProcessingML - Definice stylu založeného na odstavci

Pokud by jste potřebovali styl, který se bude zabývat jen formátováním znaků – tokem textu, stačí vytvořit styl s typem znak. Definice je podobná jako v případě odstavce, jen atribut *type* elementu *style* má hodnotu *character*. I nastavení formátování je stejné. Poněvadž se styl týká znaků, nesmí obsahovat nastavení odstavce, ale pouze jen toku textu (*rPr*). Kód 16 je ukázka stylu, pro který platí že znaky v tomto stylu budou tučné.

```
<w:style w:type="character" w:styleId="Tucny">
        <w:name w:val=" Tucny " />
        <w:basedOn w:val="Normal" />
        <w:qFormat />
        <w:rPr> <w:b /> </w:rPr>
</w:style>
```

Kód 16 WordProcessingML - Definice stylu založeného na znaku

Styly s typem tabulka jsou oproti předchozím dvěma mnohem složitější. V těchto stylech je umožněno ukládat daleko více informací než v obyčejné tabulce. Pomocí WordProcessingML můžete tabulku rozdělit do více sekcí, jako horní a dolní řádek, první a poslední sloupec nebo střídavé řádky. Definici stylu tabulky nebudu uvádět. Je to téměř shodné s předchozími typy. Jen s rozdílem atributu *type=“table“*. Pokud bude potřeba, lze v tomto stylu nastavit formátování odstavce a textu přes *pPr* a *rPr*. Co je u stylu tabulky nové oproti přechozím ukázkám, je ohraničení pomocí elementu *tblBorders* jež je potomkem elementu *tblPr.* Zápis ohraničení je úplně stejný, jako tomu bylo při přímém nastavení ohraničení buňky v ukázce Kód 11. Daleko zajímavější jsou rozšíření pro první a poslední řádek/sloupec. Tato nastavení se píší do elementu *tblStylePr*, který je přímo potomkem *style*. O jaký typ nastavení půjde, se uvede pomocí atributu *type*. Možné hodnoty jsou *firstRow* pro první řádek tabulky, *lastRow* pro poslední řádek a podobně *firstColumn* a *lastColumn* pro první a poslední sloupec tabulky. Lze také rozlišit lichý a sudý řádek v horizontálním nebo vertikálním směru pomocí *band1Horz* nebo *band2Vert*. Tato klíčová slova se mění podle toho, jestli obsahují „1“ , tak je to lichý řádek, nebo „2“ pro sudý řádek, „horz“ pro horizontální směr a „vert“ pro vertíkální směr.

```
<w:style w:type="table"
        <w:name w:val="MujStylTabulky" />
        <w:qFormat />
        <w:rPr>
                <w:color w:val="31849B" />
        </w:rPr>
        <w:tblPr>
                <w:tblBorders>
                        <w:top w:val="single" w:sz="8" w:color="4BACC6" />
                        <w:bottom w:val="single" w:sz="8" w:color="4BACC6" />
                </w:tblBorders>
        </w:tblPr>
        <w:tblStylePr w:type="firstRow">
                <w:rPr>
                        <w:b />
                        <w:color w:val="FFFFFF" />
                </w:rPr>
                <w:tcPr>
                        <w:tcBorders>
                                <w:bottom w:val="single" w:sz="8"/>
                        </w:tcBorders>
                </w:tcPr>
        </w:tblStylePr>
        <w:tblStylePr w:type="firstCol">
                <w:rPr>
                        <w:b />
                </w:rPr>
        </w:tblStylePr>
        <w:tblStylePr w:type="band1Horz">
                <w:tcPr>
                        <w:shd w:val="clear" w:color="auto" w:fill="D2EAF1" />
                </w:tcPr>
        </w:tblStylePr>
</w:style>
```

Kód 17 WordProcessingML - Definice stylu tabulky

3.5 SpreadSheetML

3.5.1 Základní struktura

Dokument SpreadsheetML v základní konstrukci musí obsahovat jeden centrální sešit (*workbook*) a samostatný soubor pro každý list (*worksheet*). S tím, že platný balíček SpreadsheetML musí obsahovat vždy sešit a nejméně jeden list, v kterékoliv složce uvnitř balíčku. Opět je potřeba soubor *[Content_Types].xml*, který obsahuje definici názvosloví pro sešit a list. Jsou to:

application/vnd.openxmlformats-officedocument.spreadsheetml.sheet.main+xml
application/vnd.openxmlformats-officedocument.spreadsheetml.worksheet+xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
        <Default Extension="bin" ContentType="application/vnd.openxmlformats-
officedocument.spreadsheetml.printerSettings" />
        <Default Extension="rels" ContentType="application/vnd.openxmlformats-
package.relationships+xml" />
        <Default Extension="xml" ContentType="application/xml" />
        <Override PartName="/workbook.xml" ContentType="application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet.main+xml" />
        <Override PartName="/ sheet1.xml" ContentType="application/vnd.openxmlformats-
officedocument.spreadsheetml.worksheet+xml" />
</Types>
```

Kód 18 SpreadSheetML - Minimální definice [Content_Types].xml

Poslední povinnou částí jsou soubory s relacemi na jednotlivé části dokumentu. Ty se ukládají do složky *_rels* v kořenové adresáři balíčku. Základní soubor, který obsahuje relace jmenuje *.rels*. V tomto souboru je například odkaz na hlavní soubor workbook.xml. Podle něj parser pozná kde hledat hlavní část dokumentu. Tam kde se vyskytuje soubour *workbook.xml* musí být opět složka *_rels* a soubor *workbook.xml.rels*, který obsahuje reference na ostatní části, které *workbook.xml* využívá, nejčastěji jsou to listy. V minimální verzi je to odkaz na první list. Pokud by list využíval další části, jako například tabulku sdílených řetězců, musí mít ve svém umístění také složku *_rels* s názvem XML souboru sešitu a přípnou *.rels*. V tomto souboru by pak byly posány refernce na ostatní části. Například na tabulku sdílených řetězců.

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
        <Relationship Id="rId1" Target="workbook.xml" Ty-
        pe="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDoc
        ument" />
</Relationships>
```

Kód 19 SpreadsheetML - Ukázka minimálního souboru .rels

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
        <Relationship Id="rId1" Target="sheet1.xml"  Ty-
        pe="http://schemas.openxmlformats.org/officeDocument/2006/relationships/workshe
        et" />
</Relationships>
```

Kód 20 SpreadsheetML - Ukázka minimálního souboru workbook.xml.rels

Prvním a hlavním úkolem souboru *workbook.xml* je nastavení jednotlivých listů dokumentu a jejich obsahu. Sešit obsahuje seznam všech listů v dokumentu. Tento seznam je vytvořen i proto, aby aplikace, která jej bude zpracovávat mohla vyjmenovat všechny listy v dokumentu. O každém listu jsou evidovány tří základní informace. Je to jméno, které je zobrazeno pro pojmenování listu, ID pro třídění listů v dokumentu a jako poslední je relační ID, které slouží k identifikaci relace a umístění souboru s definicí listu.

```
<workbook xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
xmlns:r="http:// schemas.openxmlformats.org /officeDocument/2006/relationships">
        <sheets>
                <sheet name="List1" sheetId="1" r:id="rId1" />
        </sheets>
</workbook>
```

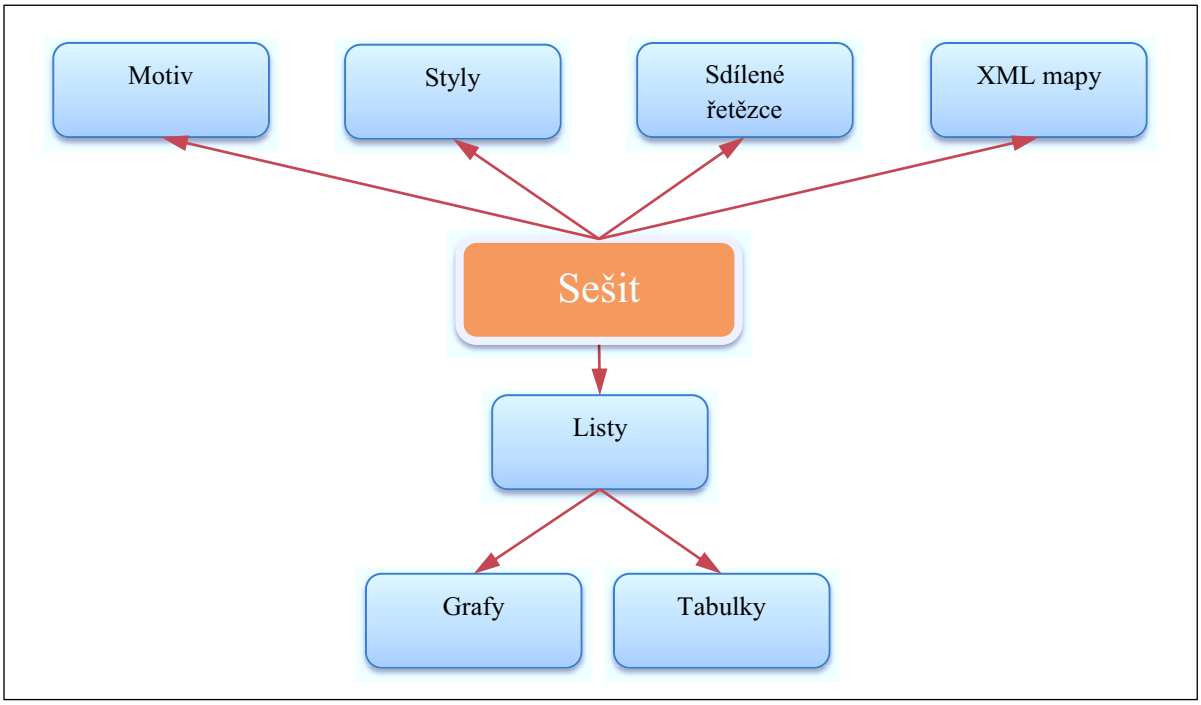
Kód 21 SpreadsheetML - Minimální definice sešitu

Každý dokument potřebuje nejméně jeden list. List je kontejner, který obsahuje data dokumentu. Data jsou uspořádána ve struktuře podobné tabulce. Abychom vytvořili prázdný list, stačí přidat pouze pár elementů. Jak je vidět v ukázce Kód 22, jako kořenový element slouží *worksheet*. Ten může obsahovat několik druhů elementů. Elementy pro nastavení vlastností dokumentu a nebo některé podpůrné elementy pro zabezpečení nebo filtrování. Nejdůležitějším elementem je bezpochyby *sheetData*, ve kterém jsou organizována data.

```
<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
        <sheetData>
                <row>
                        <c>
                                <v>ahoj</v>
                        </c>
                </row>
        </sheetData>
</worksheet>
```

Kód 22 SpreadSheetML - Ukázka minimální definice listu.

SpreadsheetML dokument může obsahovat mnoho částí. Následující obrázek „Struktura SpreadsheetML dokumentu“ popisuje většinu částí, které můžete běžně najít v dokumentech.



Obrázek 14 Struktura SpreadsheetML dokumentu

3.5.2 Tvorba listů

List obsahuje data ve formě tabulky. Pro vytvoření takové tabulky zadáte uvnitř elementu *sheetData* řádky (*rows)* a v nich buňky, které obsahují hodnotu. Element *row* definuje nový řádek. Zpravidla jsou řádky číslovany automaticky. To se dá ale obejít pomocí nastavení určitých atributů, které probereme později. Uvnitř řádku se definují buňky, které se značí elementem *c* a jejich hodnota, která v nejjednoduší variantě je buď text (element *t*) nebo číselná hodnota (element *v*).

Pokud jste, ale vytvořili sešit přímo v Microsoft Excel, jsou textové hodnoty buňky uloženy odlišným způsobem. Používají takzvané sdílené řetězce. Tato optimalizace snižuje velikost výsledného souboru. Pokud ale chcete textovou hodnotu uložit klasickým způsobem, musí buňka obsahovat element *is*, který je zkratkou inline-string. A uvnitř tohoto elementu teprve vložit textový element *t.* (viz. Kód 23 SpreadsheetML - Ukázka textové a číselné buňky.)

```
<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
  <sheetData>
    <row>
      <c t="inlineStr">
        <is>
          <t>ahoj</t>
        </is>
      </c>
      <c>
        <v>157</v>
      </c>
    </row>
  </sheetData>
</worksheet>
```

Kód 23 SpreadsheetML - Ukázka textové a číselné buňky.

3.5.3 Vzorce

Největší síla SpreadsheetML spočívá právě ve vzorcích. Vzorce umožňují automatickou kalkulaci založenou na hodnotách odkazovaných buněk. Vzorce jsou uloženy přímo v buňce, která obsahuje vzorec. Vzorce jsou uloženy jako textové řetězce v elementu *f*. Obsahují navíc element *v*, který v předchozím příkladu ukládal číselnou hodnotu. Nyní však ukládá poslední známý výsledek vzorce. Není to sice povinné, ale významně to urychluje otevírání SpreadsheetML dokumentu, protože se nemusí čekat, než se přepočítají hodnoty všech vzorců v sešitu. Pokud se ukládají lokalizované funkce, jako je například česká SUMA, uloží se v internacionální podobě, aby se předešlo problémům s kompatibilitou.

```
<c>
        <f>A3 * (B3/100+1)</f>
        <v>119</v>
</c>
```

	A	B	C
1			
2	cena	dph	s dph
3	100	19	119

Kód 24 SpreadSheetML - Definice rovnice

3.5.4 Optimalizace sešitu

Aby nebyla výsledná velikost SpreadSheetML souboru příliš velká, obsahuje specifikace několik možností, které minimalizují zápis XML souborů. Asi nejvýznamnější je optimalizace pro tabulky, kdy se nemusí definovat všechny řádky a buňky, ale jen ty které potřebujete.

3.5.4.1 Optimalizace řádků a buněk

Až doposud se řádky a buňky počítali od levého horního rohu. Pokud bychom chtěli přidat text třeba do buňky na pozici C6, museli bychom definovat zbytečné množství řádků a buněk. Díky optimalizaci stačí definovat pouze jeden řádek a jednu buňku. Stačí řádku a buňce přidat atribut označující pozici řádku nebo buňky. Ten se značí *r*. Pokud jde o řádek obsahuje pouze číslo řádku, ale v případě buňky obsahuje jak písmeno určující sloupec i číslo řádku.

```
<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
   <sheetData>
      <row r="5">
         <c r="E5">
            <v>157</v>
         </c>
      </row>
   </sheetData>
</worksheet>
```

Kód 25 SpreadsheetML - Optimalizace kódu pro zápis řádku a buněk.

3.5.4.2 Tabulka sdílených řetězců

Optimalizovat se dají i textové řetězce v sešitu. Pokud sešit obsahuje více stejných textových řetězců je bezpochyby lepší využít takzvanou tabulku sdílených řetězců. Tato tabulka obsahuje seznam všech textových řetězců v sešitu a jejich jednoznačnou identifikaci. Na místo textových hodnot jsou pak v příslušných místech jen odkazy odpovídající požadovanému řetězci v tabulce sdílených řetězců. Tabulka sdílených řetězců je samostatná část sešitu a je společná pro všechny listy. Ukládá se jako samostatný XML soubor, jehož typ je *application/vnd.openxmlformats-officedocument.spreadsheetml.sharedStrings+xml* ten musí být navíc patřičně zapsán v souboru *[Content_Types].xml*. Navíc je třeba přidat odkaz do retenčního souboru *.rels* kde bude jako typ obsahu uveden *http://schemas.openxmlformats.org/officeDocument/2006/relationships/sharedStrings*

Tabulka sdílených řetězců se značí elementem *sst*. Jednotlivé položky tabulky se uvádí v elementu *si*, což znamená string-item.

```
<sst xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
      <si>
            <t>Jan Novak</t>
      </si>
      <si>
            <t>Petr Adamek</t>
      </si>
</sst>
```

Kód 26 SpreadsheetML - Tabulka sdílených řetězců

Tímto jsme nadefinovali tabulku sdílených řetězců. Ještě je ale musíme nahradit odkazy v tabulkách sdílených řetězců přímo v buňkách. U textu uloženého přímo v buňce se používal atribut *t* udávající typ řetězce jako inlineStr a element *is* obsahující text na řádku. Při odkazování do tabulky sdílených řetězců je zápis trochu jiný. Jako typ *t* se uvádí *s* jako sdílený řetězec. Element *is* zde není ani nemá žádnou obměněnou verzi. Místo toho se používá element *v* pro ukládání číselných hodnot. Číslo uložené v něm odpovídá indexu v tabulce sdílených znaků. Podle naší ukázky by to byl index 0 pro Jana Nováka a index 1 pro Petra Adámka.

```
<row>
        <c  t="s">
                <v>0</v>
        </c>
        <c  t="s">
                <v>1</v>
        </c>
</row>
```

Kód 27 SpreadsheetML - Použití odkazu do tabulky sdílených řetězců.

3.5.5 Formátování obsahu

Jako jiné části Open XML můžete i pro SpreadsheetML vytvořit styly, které jsou definovaný v samostatném souboru uvnitř balíčku. Postup je podobný jako v kapitole 3.4.5. Struktura souboru obsahující styly je trošku jiná, než tomu bylo pro WordProcessingML. Zde k dispozici daleko více elementů pro stylování různých částí a maximální znuvupoužitelnosti jednotlivých částí.

Definice stylu pro SpreadsheetML probíhá trochu odlišným způsobem. Styly se ukládájí opět do souboru *styles.xml*, který je referencován v *workbook.xml.rels*. Základ tvoří čtyři elementy: *fonts, fills, borders* a *numFmts*. Jak název napovídá element font slouží k definici fontů, *fills* definuje výplně, *borders* ohraničení a *numFmts* definuje format čísel. Tyto elementy neformátují vzhled buňky přímo, ale pouze definují různé varianty (sady nastavení), ze kterých se sestavují komplexní styly pro buňku. V těchto elementech jsou definována jednoznačná nastavení podle typu elementu. Například element *fonts* obsahuje difinici všech fontů, které mohou být použity. Jednotlivé styly pak použijí font tak, že se odkáží na index záznamu v elementu *fonts*.

Element *borders* urdržuje seznam ohraničení. Každé ohraničení je definováno jedním elementem *border*). Uvnitř můžete definovat šest ohraničení pomocí vyhrazených typů jako jsou *left* nebo *bottom*. Šířku ohraničení není možné určit přímo, místo toho se určí typ ohraničení. Například jednoduché (*thin)* nebo dvojité (*double)*. Barva ohraničení je definována pomocí ARGB zápisu (Alfa, Red, Green, Blue)

```
<borders>
        <border>
                <bottom style="double">
                        <color rgb="FFAAC108" />
                </ bottom >
        </border>
        … další ohraničení
</borders>
```

Kód 28 SpreadSheetML - definice listu ohraniční v styles.xml

Nastavení písma je podobné jako tomu je u WordProcessingML (viz kapitola 3.4.5).

```
<fonts>
        <font>
                <sz val="16" />
                <name val="Arial" />
                <color rgb="00FF02BB" />
        </font>
        … další fonty
 </fonts>
```

Kód 29 SpreadsheetML - definice listu písem v styles.xml

Výplně definují typ výplně, barvu písma a barvu pozadí.

```
<fills>
        <fill>
                <patternFill patternType="solid">
                        <bgColor rgb="FF000000" />
                      <fgColor rgb="FFFFFFFF" />
                </patternFill>
        </fill>
        … další výplně
</fills>
```

Kód 30 SpreadsheetML - definice listu výplní v styles.xml

Máme dvě možnosti, jak nastavit formát buňce. Můžeme nastavit globální styl, jako je například "normální" nebo můžeme nastavit styl pro konkrétní buňku nebo buňky. Globální definice probíhá v elementu *cellStyleXfs*. Většinou se ale používá definice konkrétního stylu. Tyto styly jsou obsažené v lementu *cellXfs*. Ten obsahuje přímo definice stylů. Nastavní konkrétních hodnot formátování probíhá pouze odkázáním se na index záznamu do příslušného ze čtyř základních elementů.

```
<cellStyleXfs count="1">
        <xf num fontId="0" fillId="0" borderId="0" />
</cellStyleXfs>
<cellXfs count="5">
        <xf fontId="0" fillId="0" borderId="0" xfId="0" />
        <xf fontId="1" fillId="2" borderId="1" xfId="0" />
</cellXfs>
```

Kód 31 SpreadsheetML - definice stylů v styles.xml

Pokud buňka má nastaven styl odkazuje se na index stylu, který jí má být nastaven.

```
<row>
        <c s="1">
             <v>0</v>
        </c>
</row>
```

Kód 32 SpreadsheetML - definice stylů v buňce

3.6 Přínos a kritika Open XML

Přínos spočívá hlavně v tom, že umožňují rychle vytvářet dokumenty z různých zdrojů dat. Výměnu dat mezi aplikacemi dat a podnikovými systémy. A oproti binárním souborům, přináší tu výhodu, že lze dokumenty snadno upravovat nebo vytvářet bez použití aplikací office. Velkou výhodou je také snížení velikosti výsledného souboru až o 75% oproti binárnímu souboru. Podstatnou výhodou je zpětná kompatibilita se staršími systémy office od verze Microsoft Office 2000, kde stačí nainstalovat pouze aktualizaci pro podporu Open Xml.

Open Xml je z mnoha stran kritizován. Zejména ze strany W3C, kterému se nelíbí, že formáty nejsou neutrální. Pro ukládání vektorové grafiky používá DrawingML místo SVG, které W3C doporučuje. A trnem v oku je také OfficeMathML, který slouží k zápisu matematických formulí, i když se doporučuje používat MathML. Dalším nestandardním přístupem ze strany Microsoft je, že v souborech používá makro-jazyky, které nejsou ve standardu definovány.

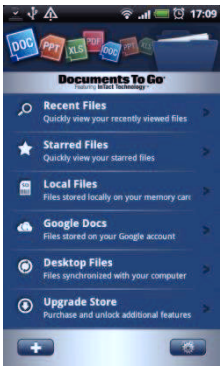
4 Aplikace pracující s Open XML

Aplikací pracujících s Open Xml v operačním systému Android existuje opravdu jenom několik. Pro běžného uživatele ale mají jednu nevýhodu, nejsou zdarma. Navíc mobilní zařízení se podstatně liší od klasických počítačů, které mají klávesnici a myš. Proto aplikace musí být přizpůsobeny těmto faktům a nemohou nabízet stejné možnosti a výhody jako desktopové aplikace. Aplikace, které jsem testoval, obsahovaly různé podmnožiny standardu Open XML, ale bohužel většinou jen ty základní. Jednotlivé části standartu, které aplikace podporují, jsou velmi rozdílné. V podstatě by se dalo říct, že neexistuje žádná univerzální aplikace, která by se dala označit za nejlepší a která by uměla stejné věci jako ostatní.

4.1 Srovnání existujících aplikací

4.1.1 DocumentsToGo 3.0

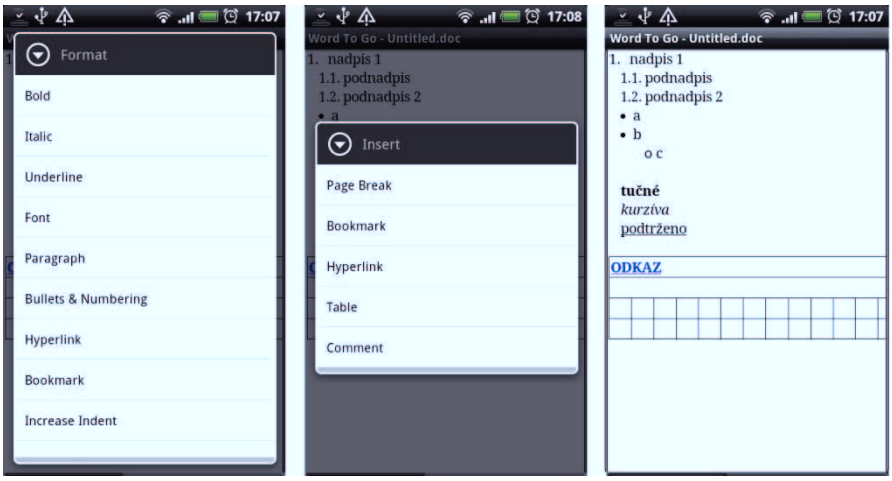
DocumentsToGo od společnosti DataViz je jedna z nejoblíbenějších a úspěšných kancelářských sad. Umožňuje čtení, editování a vytváření souborů kancelářského balíku Microsoft Office (Microsoft Word, Excel a PowerPoint). Kromě toho ji lze použít i ke čtení PDF souborů. K dostání ve dvou verzích. Standardně jako lite verze, která umí dokumenty pouze číst. Ale v přepočtu za asi 250 Kč si lze koupit klíč, který umožní editaci a vytváření dokumentů. DocumentsToGo je sice nejdražší placenou aplikací pro zpracování Open XML, ale zase nabízí nesrovnatelně víc funkcí než konkurenční aplikace.



Obrázek 15 DocumentsToGo - hlavní obrazovka

WordToGo 3.0

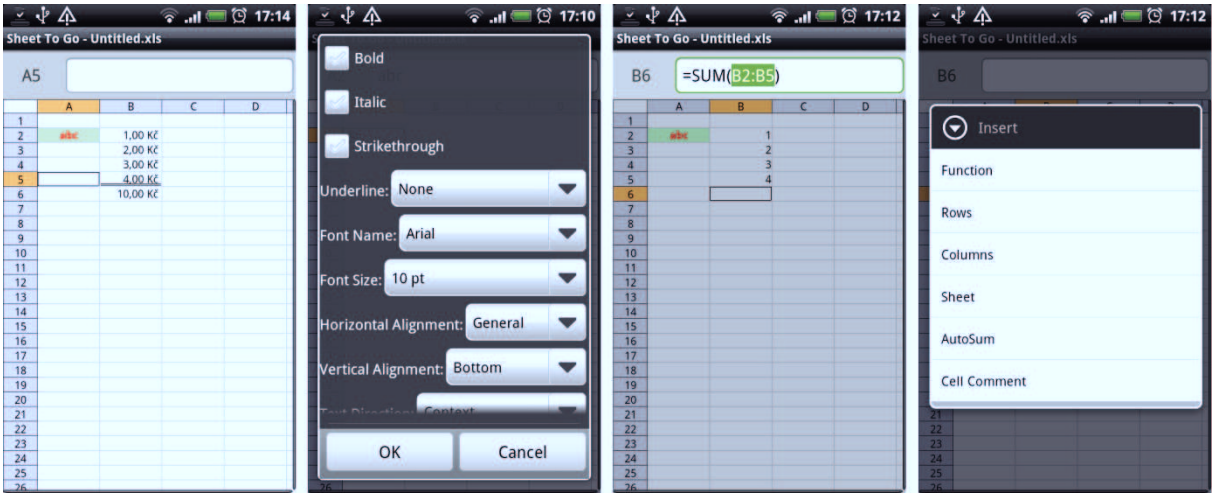
WordToGo nabízí asi největší množství funkcí pro práci s WordProcessingML v prostředí Androidu se kterými jsem se setkal. Mimo základní formátování textu nabízí dokonce víceúrovňové seznamy odkazy nebo tabulky. Práce s programem je komplikovanější, protože postrádá nástrojovou lištu a veškeré operace včetně formátování textu se musí provádět přes menu.



Obrázek 16 Snímky obrazovek WordToGo

ExcelToGo 3.0

Opět jedna z nejlepších aplikací pro práci se SpreadSheetML. Nabízí široké možnosti. Včetně formátování a zarovnání textu v buňce. Lze používat alespoň základní funkce. Ovládání veškerého nastavení probíhá opět přes menu.



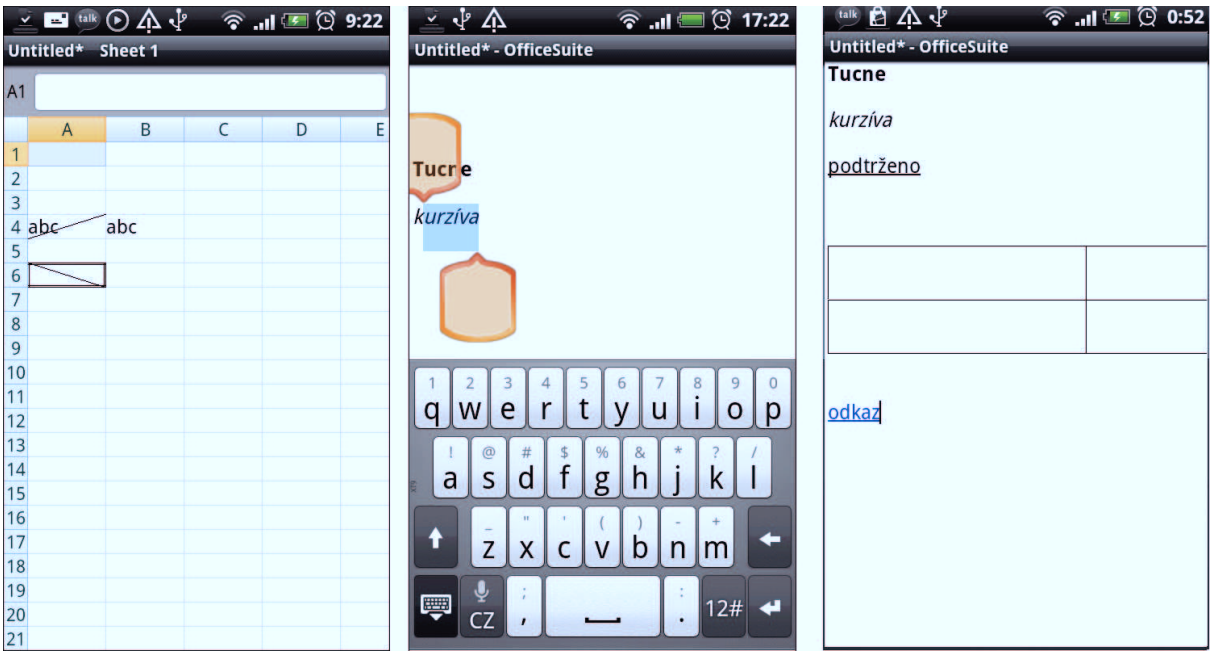
Obrázek 17 Snímky obrazovek ExcelToGo

4.1.2 OfficeSuite

OfficeSuite je další z povedných implementací, která umí pracovat s Open XML. Zdarma je bohužel jen 30ti denní zkušební verze. Po uplynutí této doby si musíte zakoupit plnou verzi, která stojí přibližně 250 Kč.

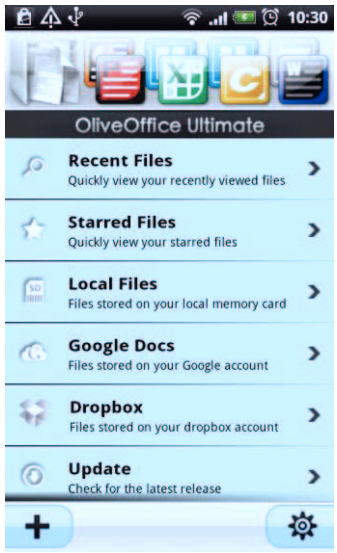
Editor pro dokumenty WordProcessingML je rovněž velice povedený a nabízí mnoho možností formátování textu. Také dokáže pracovat s tabulkami. Mimo jiné umí nastavit široké vlastnosti odstavce včetně řádkování nebo odsazení, více úrovňové číslování nebo zalomení sekcí. Opět se jedná o velmi kvalitní editor s nepřeberným množstvím funkcí.

Tabulkový procesor je jednou z nejvydařenějších implementací z porovnávaných balíků. Rozhraní je velice rychlé a nabízí nejobsáhlejší možnosti formátování, ohraničení buňky nebo zarovnání textu. Bohužel jsem nikde nenašel možnost jak nastavit formátování nebo barvu v buňce. Jinak by to byl bezpochyby nejobsáhlejší editor, který jsem testoval.



Obrázek 18 Snímky obrazovek OfficeSuite

4.1.3 OliveOffice

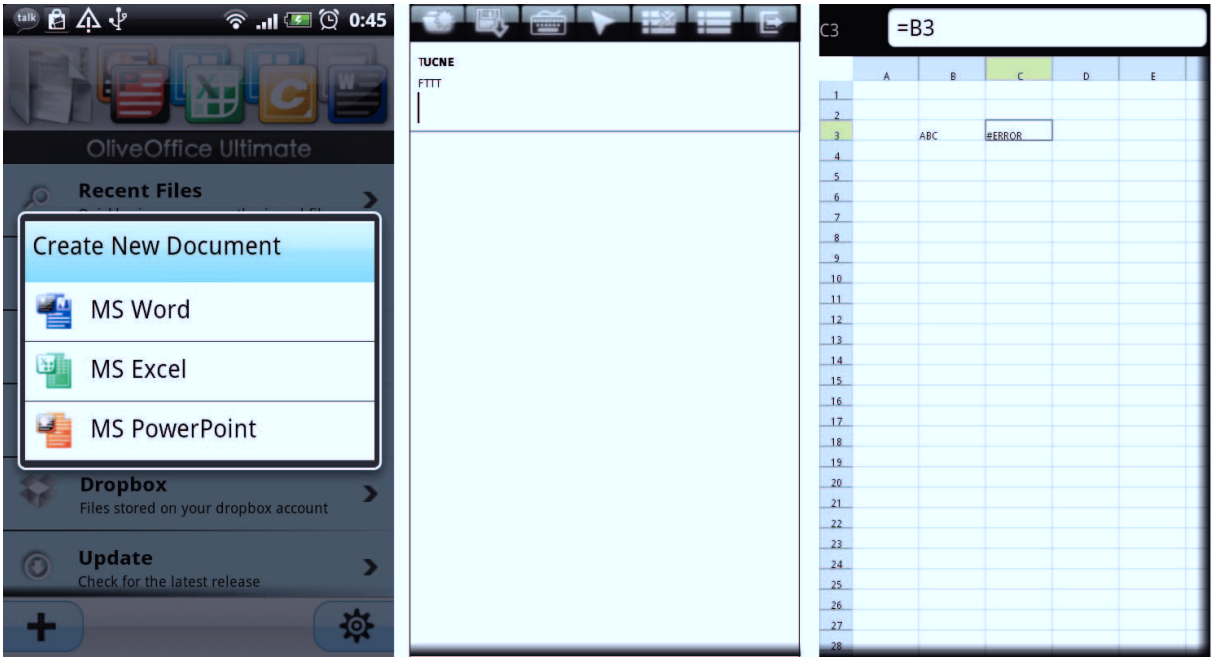


Obrázek 19 OliveOffice - hlavní obrazovka

Ve verzi Basic opět umožňuje jen prohlížení dokumentů Word, Excel, prezentací, PDF nebo souborů nápovědy *.chm*. Má však ještě spoustu nedostatků, někdy se může stát, že soubory Excel nejdou otevřít nebo je rozhozené formátování dokumentů. Nedávno však byla uvolněna verze Premium, která zdarma umožňuje i vytvářet dokumenty, ale jen s několika funkcemi v porovnání s DocumentsToGo. SpreadSheetML podporuje pouze některá nastavení ohledně fontu, formátu buněk. Nepodporuje žádné funkce, ani ty základní, jako jsou reference na jiné buňky. A popravdě aplikace často havaruje i při běžné práci. Editor WordProcessingML také nemá mnoho funkcí, lze jen nastavit formátování textu a pracovat se schránkou. Pokud chceme nastavit formátování určité části textu, je třeba kliknout na ikonku výběru a pracně označit požadovaný úsek a až po té přes menu nastavit formátování. Ale pochvalu si zaslouží nástrojová lišta, která umožňuje například zobrazení a skrytí klávesnice.

Editor pro sešity Excel má ještě méně funkcí, ale s více chybami. Jediné, co lze v tomto editoru psát do buňky, je čistý neformátovaný text. Nefungují ani reference na jiné buňky. A hned při prvním stupni zvětšení se zneviditelnil veškerý napsaný text.

OliveOffice sice nemá mnoho funkcí, ale je k dispozici zdarma. A jako skromný editor pro nenáročného zákazníka se dá použít. A časem snad i dožstihne svou konkurenci.



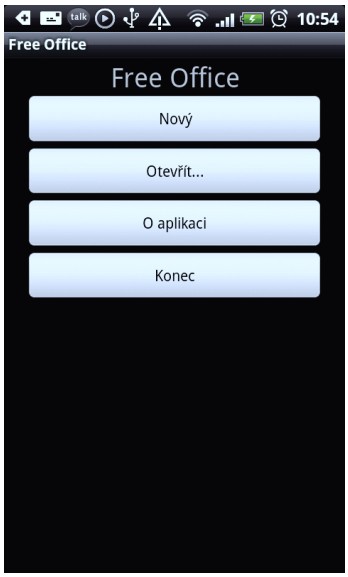
Obrázek 20 Snímky obrazovek OliveOffice

5 Vlastní aplikace pro práci s Open XML

5.1 Popis aplikace

Součástí práce je i vlastní implementace kancelářského balíku pro operační systém Android. Mým cílem je navrhnout aplikaci, která bude zdarma a jednoduše použitelná pro každého člověka. Při implementaci programu jsem se snažil využít poznatků z práce se systémem Android a zároveň se i poučit z problémů konkurečních aplikací. Snažil jsem se navrhnout jiná řešení, která by umožnila uživateli jednodušší ovládání a pohodlnější práci s aplikací. Avšak vývoj komplexní aplikace, která by obsahovala veškerou funkčnost, kterou nabízí konkurence je ale velice nákladná a časově náročná. Proto aplikace demonstruje jen základní funkce Open XML.

Aplikace umí otevírat a vytvářet textové dokumenty a sešity Open XML. Balík je navržen jako jedna aplikace s centrálním menu, kde si uživatel zvolí, jestli chce otevřít nebo vytvořit dokument. Podle jeho volby se pak načte příslušné rozhraní buď pro textové dokumenty nebo sešity. Tato rozhraní budou posána v následujících kapitolách.



Obrázek 21 Vlastní implementace - úvodní menu.

5.2 Implementace WordProcessingML

Editor pro WordProcessingML slouží k zobrazení i vytváření dokumentů. Uživatelské rozhraní je jednoduché a skládá se z nástrojové lišty pro snadnou práci s textem a prostoru pro psaný text.

Jako centrální část editoru jsem použil textové pole, které obsahuje formátovaný text. Na rozdíl od obyčejného textového pole ale umí používat různé formátování textu. Výsledný text je reprezentován jako kolekce intervalů (*spans*) a každý *span* obsahuje své formátování. Lze použít různé druhy pro nastavení intervalu. Například nastavit třeba tučný, kurzivní nebo podtržený interval nebo jejich kombinaci. Ale existují i speciální intervaly pro barvy pozadí a popředí nebo pro velikost textu. Dokonce třeba pro odrážkovaný text. Avšak toto pole má své limity a nebyl jsem schopen ho použít například pro vykreslení tabulky nebo zobrazení obrázku. Ale možné to určitě je, protože třeba WordToGo (viz. 4.1.1 DocumentsToGo 3.0) umí používat i tabulky.

Obrázek 22 Vlastní implementace - editor WordProcessingML

Editor samozřejmě neobsahuje všechny možnosti jako desktopová verze, ale snažil jsem se obsáhnout alespoň ty základní, které by neměli chybět při práci s textovým editorem. Následující seznam zobrazuje podporované a nepodporované funkce editoru.

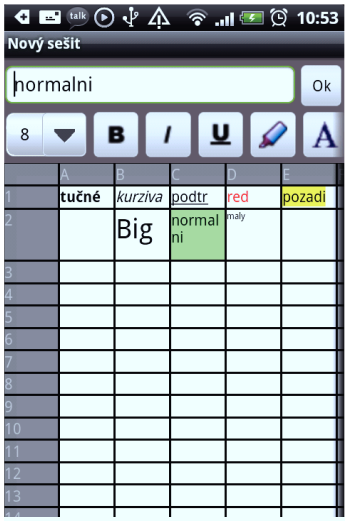
- ✓ Zobrazení odstavců
- ✓ Formátování textu – tučné, kurzivá a podtržené
- ✓ Barva písma
- ✓ Barva zvýraznění textu
- ✓ Nastavení velikosti textu
- × Obrázky
- × Tabulky
- × Odrážky a číslování

I přesto, že editor neobsahuje všechny funkce jako konkureční aplikace, je pro běžné použití dostačující. A podle zkušeností můžu říct, že nabízí nadstandartní funkce textového editoru v prostředí Android.

5.3 Implementace SpreadsheetML

Stejně jako editor WordProcessingML umí vytvářet a zobrazovat soubory, tak i editor pro SpreadsheetML umí to stejné. Oproti textovému editoru obsahuje dvě nástrojové lišty, které jsou umístěny pod sebou. Pod nástrojovými lištami je hlavní prostor, který obsahuje sloupce a řádky pro práci s listem sešitu.

Jednotlivé buňky jsou řešeny jako textové popisky a nikoliv jako editovatelná textová pole. Je to hlavně zdůvodu, že když by se použila textová pole, tak by se při každém výběru buňky zobrazila virtuální klávesnice, která by výrazně zmenšila prostor editoru. Proto jsem raději zvolil použití horní nástrojové lišty, která nastaví text aktivní buňce. V praxi to funguje tak, že pokud uživatel chce nastavit text nějaké buňce, kliknutím na ni ji označí jako aktivní. Její hodnotu pak změní v textovém poli v horní nástrojové liště a potvrdí tlačítkem ok.



Obrázek 23 Vlastní implementace - editor SpreadsheetML

Další nástrojová lišta pak slouží k přímému formátování obsahu buňky stejně jako ve Wordu. Obsahuje navíc tlačítko pro vložení součtu rozsahu buněk. Po stisknutí tlačítka se zobrazí dialog, kde uživatel zadá rozsah buněk a po potvrzení tlačíkem se do označené buňky spočítá hodnota součtu.

Pokusil jsem se naimplementovat alespoň základní funkce, které se týkají zejména formátování obsahu buněk. Narazil jsem ale i na několik omezení, které mi znemožnili vykreslení ohraničení. Některé pokročilé funkce, jako jsou třeba vzorce, jsem zatím neimplementoval. Například vzorce pro odkazování se na jiné hodnoty jiných buněk. Implemetace by byla komplikovanější o to, že bych mu-

sel udržovat pro každou buňku seznam buněk, které se ve vzorci odkazují na její hodnotu, a při změně hodnoty projít seznam navázaných buněk a přepočítat jejich hodnotu. Hlavním důvodem neimplementace jsou zejména problémy s uživatelským rozhraním, které už tak umí zobrazit jen pár řádků a sloupců s velkým vytížením systému. Avšak editor umí zobrazit při otevírání dokumentu v buňce místo vzorce alespoň předpočítané hodnoty funkcí.

Následující seznam popisuje, které funkce jsem implementoval a které ne.

- ✓ Formátování textu v buňce – tučné, kurzivá a podtržené.
- ✓ Barva písma buňky
- ✓ Barva pozadí buňky
- ✓ Vzorce – pro ukázku jen suma.
- × Ohraničení buňky
- × Odkazy na ostatní buňky

5.4 Otevírání a ukládání souborů

Při otevírání nebo ukládání dokumentů se na externím uložišti vytvoří dočasná složka s jedinečným názvem. Při ukládání dokumentu se do této dočasné složky vytvoří všechny potřebné soubory balíčku pro dokument. Jakmile je vytvořen poslední soubor, je předáno volání třídě ZipWorker, který zabalí obsah dočasné složky do zip archívu s příponou podle typu ukládaného souboru.

Postup při otevírání dokumentu je podobný. Opět se vytvoří složka na externím úložišti, do které je rozbalen obsah otevíraného souboru. Pro parser jsou nejdůležitější *.rels* soubory ve složce *rels*. Napřed se podívá do souboru *.rels* kde najde referenci na hlavní část dokumentu a cestu, kde ji najde. Předává řízení metodě pro zpracování hlavního dokumentu, která si podle dodatečného *rels* souboru pro hlavní část najde umístění dalších částí dokumentů pro zpracování. Po zpracování všech potřebných souborů zobrazí jejich obsah do příslušných souborů a smaže dočasnou složku.

5.5 Problémy a komplikace

5.5.1 Nástrojová lišta

Ve své implementaci jsem se rozhodl vzít do úvahy zkušenosti z porovnání již existujích balíků, kde mi přišlo ovládání aplikace příliš složité. Veškeré úkoly se museli dělat většinou přes menu. I takové banality jako je třeba nastavení tučného písma se stávályi titěrnou prací, která obsahovala podle mého názoru až příliš mnoho úkonů. I jednoduché úkoly, jako je psaní formátovaného textu, se nedalyi udělat žádnou přímou cestou. Princip změny formátování spočíval v ručním označení textu a nastavení přislušeného formátu přes menu. Asi nejobtížnějším úkolem bylo ručně označit úsek textu, ve kterém potřebuji nastavit nějaký formát, zejména pokud jsem chtěl označit jen část slova. V podstatě žádná z porovnávaných aplikací nedisponovala volbou přímého psaní v nastaveném formátu.

Jako nejsnažší řešení se mi jevilo vytvořit nástrojovou lištu podobnou jako má Microsoft Word.

Obrázek 24 Vlastní implementace - nástrojová lišta pro rychlé formátování.

Překvapilo mě, že žádný konkureční program takovou lištu nemá. Řešení jsem si představoval tak, že tlačítka v nástrojové liště budou manipulovat s proměnnými, které budu ukládat nastavení příslušného tlačítka a při stisknutí znaku na klávesnici se znak vysází podle nastavení proměných. Hned při prvním pokusu o návrh jsem ale narazil na problémy, které se nejprve jevily jako nevyřešitelné. Prvním z nich bylo, že Android neumožňuje odchytávání znaků napsaných na virtuální klávesnici. Moje představa byla taková, že budu sázet jednotlivé znaky podle nastavení v proměnných. V podstatě jediný možný způsob na sledování změny v textovém poli je použití třídy TextWatcher, která je posluchač události TextChanged textového pole. TextWatcher sleduje změny textu v textovém poli a při každé změně textu se zavolají metody v této třídě. Tohle řešení by fungovalo, ale mělo jednu vadu. Pokaždé, když jsem v metodě TextWatcheru upravil hodnotu textu, tak se zavolala znovu metoda oznamující změnu textu a program skončil v nekonečné smyčce. To je možná příčinou proč konkurence nepoužívají nástrojovou lištu. Také jsem to napřed vzdal a přemýšlel o tom, že přeci jen budu muset nastavovat formátování přes menu. Ale napadla mě ještě jedna myšlenka, která se osvědčila. Klíč byl v tom, že jsem si udělal vlastní třídu odvozenou z TextWatcher která obsahovala referenci na textové pole na které byla navázána. Abych zabránil nekonečné smyčce při změně textu, využil jsem odkazu na textové pole a odebral mu posluchače události TextChanged. Díky tomu jsem si mohl změnit text podle svých potřeb a nakonec zase přidat posluchače události na sebe sama. Tím se eliminoval problém se zacyklením a mohl jsem už nástrojovou lištu používat k různým formátováním textu.

5.5.2 Odstavce

Při tvorbě editoru WordProcessingML jsem narazil na komplikaci, že textové pole sice umí různé formátování textu, ale neumí zobrazit odstavce. Musel jsem se tedy obejít bez zarovnání textu v odstavci nebo zobrazení případných mezer před a za odstavci. Místo toho jsem pro oddělení odstavců použil prázdný řádek.

5.5.3 Ohraničení buněk

Buňky v editoru SpreadsheetML jsou obyčejná pole pro zobrazení textu. Po delším prohledávání internetových diskuzí ohledně toho, jak udělat ohraničení okolo takového pole, jsem narazil na to, že Android vůbec neumí vykreslit ohraničení pro popisek. Abych, aspoň vizuálně oddělil řádky a sloupce v editoru a vykreslil mřížku mezi nimi, musel jsem použít trik v tom, že pozadí uživatelského ohraničí je černé a popisky mají mezi sebou nastavený rozestup 1dp. To opticky vytvoří mřížku mezi buňkami, ale znemožní to jakékoliv nastavení ohraničení buňce.

5.5.4 Rychlost uživatelského rozhraní

Při práci s editorem SpreadsheetML se objevila komplikacei v rychlosti načtení uživatelského rozhraní a práci s ním. Pokud editor obsahoval příliš mnoho buněk zobrazení uživatelského rozhraní trvalo příliš dlouho. Při posunutí sešitu do strany byl pohyb velice trhaný a zdlouhavý. Proto jsem musel

omezit prostředí pro vytváření dokumentu jen na určitý počet sloupců a řádků, tak aby se dalo s rozhraním alespoň trochu pracovat. Při otevírání dokumentu, ale systém vykreslí všechny řádky a sloupce, které jsou v něm definovány. Komplikace mohou ale nastat při obsáhlých dokumentech. V takových případech se může aplikace stát neovladatelnou. Jednak buňky obsáhnou velký prostor v operační paměti. A i malé posunutí uživatelského rozhraní zaměstná procesor na plný výkon po dlouhou dobu.

5.6 Shrnutí

I přes některé technologické komplikace, se kterými jsem se setkal, si myslím, že se mi podařilo napsat kvalitní textový editor, který má potenciál pro široké použití. Když bych ho chtěl použít jako alternativu k poznámkovým blokům, které jsou zdarma pro Android, tak nabízí nadstandartní funkce editoru. Kromě toho považuji nástrojovou lištu za největší přednost mezi ostatními editory, protože velice usnadňuje práci s editorem a jedoduché ovládání je určitě přínosem pro práci uživatele s aplikací.

U editoru pro SpreadsheetML se mi sice podařilo implementovat alespoň základní funkce tohoto formátu, a vzhledem k tomu jaké má editor problémy s načítáním obsahu a vykreslením buněk, jsem byl donucen omezit počet sloupců a řádků na listu při tvorbě nového dokumentu. To velice omezuje použitelnost editoru. Přestože většinu jednodušších dokumentů dokáže zobrazit, tak by si editor zasloužil ještě další zkoumání a navrhnout lepší způsob, jak vykreslit buňky.

6 Závěr

Je třeba si uvědomit, že rozdíl mezi ovládáním klasického PC a mobilního zařízení s dotykovou obrazovkou je obrovský a proto není možné realizovat veškerou funčnost Open XML v prostředí mobilního zařízení. Ale určitě lze obsáhnout alespoň základní funkce standardu, které jsem demonstroval při návrhu vlastní aplikace.

Vývoj aplikací pro zpracování WordProcessingML a SpreadsheetML mi přinesl nové zkušenosti při vývoji pro mobilní zařízení, zejména Android. Doposud jsem pracoval pouze s prostředím pro vývoj pro Windows Mobile. Když porovnám vývoj pro obě platformy, tak musím říct, že vývoj pro Android mě zaujal daleko více. Jako začátečník jsem měl ale problémy najít kvalitní zdroje a příklady pro vývoj. Google sice provozuje stránky podobné MSDN, ale nejsou tam kvalitní ukázky použití, spíše obsáhlé definice a reference jiné třídy. Ale i tak se dalo po internetových diskuzích najít dobré příklady.

Během vývoje se mi podařilo vytvořit snadné uživatelsky přívětivé ovládání editorů, které nemá ani z žádných porovnávaných balíků. Pokud bych věnoval vývoji a testování další čas, myslím si, že by se mnou vytvořená aplikace mohla klidně použít jako volně dostupný balík pro práci s Open XML.

Literatura

1. Rice, Frank. *Introducing the Office (2007) Open XML File Formats* [Online] 13.3.2011. [Citace: 13.3.2011] <http://msdn.microsoft.com/en-us/library/aa338205(office.12).aspx>.

2. *Android (operační systém) – Wikipedia, the free encyclopedia*[Online] 13.3.2011. [Citace: 13.3.2011] <http://cs.wikipedia.org/wiki/Android_(opera%C4%8Dn%C3%AD_syst%C3%A9m)>

3. Surhone M., Lambert. *Office Open XML File Formats*. BetaScript Publishing ISBN 978-3639952650

4. Surhone M., Lambert. *Comparison of Office Open XML and OpenDocument*. BetaScript Publishing ISBN 978-6133033856

Seznam příloh

Příloha č. 1: Dokument vize

Příloha č. 2: Software Architecture Document

Adresářová struktura přiloženého DVD

/Aplikace	Verze aplikace, která jde spustit (nainstalovat) na zařízení s OS Android
/Src	Adresář obsahuje zdrojové kódy aplikace
/Prilohy	Adresář obsahuje přílohy k práci
/texty	soubory s textem práce, zadání, klíčová slova a abstrakt

Kancelářský balík FreeOffice

FreeOffice

Dokument vize

Příloha č. 1

Verze <1.3>

Historie revizí

Datum	Verze	Popis	Autor
20. 12. 2010	1.0		Bc. Pavel Bednář
15. 02. 2011	1.1		Bc. Pavel Bednář
18. 04. 2011	1.2		Bc. Pavel Bednář
30. 04. 2011	1.3		Bc. Pavel Bednář

Obsah

Vize

1. Úvod

1.1 Účel dokumentu

Účel tohoto dokumentu je shrnout všechny informace potřebné na vytvoření system, který splňuje všechny požadavky na funkčnost zadavatele.

1.2 Rozsah

Tento dokument popisuje všechny zainteresované osoby a funkční/nefunkční požadavky na systém.

2. Popis uživatelů

Tato sekce definuje uživatele systému a popisuje jejich roli.

2.1 Souhrn uživatelů

Jméno	Popis	Používá
Běžný uživatel	Člověk, který pracuje s aplikací.	Funkce

2.2 Profily uživatelů

2.2.1 Běžný uživatel

Představitel	Někdo kdo pracuje s aplikací
Popis	Potřebuje vytvářet a otevírat dokumenty Open XML

3. Souhrn produktu

3.1 Perspektiva produktu

Tento systém je nezávislý na okolních systémech. Produkt dodržuje standart zápisu jazyků Open XML.

3.2 Souhrn schopností

Hlavním úkolem systému je vytváření, editace a otevírání dokumentů WordProcessingML a SpreadsheetML, které jsou součástí specifikace Open XML.

4. Vlastnosti produktu

Systém umí pracovat s dokumenty Open XML

- Uživatel může vytvářet dokumenty
  - *Popis: Uživatel si chce vytvořit textový dokument. Mělo by se mu zobrazit rozhraní pro pohodlné psaní dokumentu s možností funkcí formátování textu. Uložený dokument musí odpovídat specifikacím pro daný typ dokumentu.*
- Uživatel může editovat dokumenty
  - *Popis.: Uživatel může také chtít upravit již existující dokument. Systém by měl být schopen zobrazit i dokumenty vytvořené v jiných aplikacích, které dodržují standart Open XML*
- Uživatel může otevírat dokumenty
  - *Popis.: Systém by měl také zobrazit již existující dokument. Systém by měl být schopen zobrazit i dokumenty vytvořené v jiných aplikacích, které dodržují standart Open XML*

4.1 Editor WordProcessingML

Editor umí formátovat text tučným písmem

- Uživatel může formátovat označený úsek textu tučným písmem
  - *Popis: Uživatel chce mít část textu zvýrazněnou tučným písmem. Systém by měl umožnit označenému úseku textu nastavit tučné písmo.*
- Uživatel může zapnout trvalé psaní tučným písmem
  - *Popis.: Uživatel chce, aby vkládaný text byl přímo tučný a nemusel jej později označovat a nastavit tučné písmo dodatečně. Systém by měl obsahovat tlačítko, které sepne psaní tučným písmem, tak aby každý nově vložený znak byl tučný.*

Editor umí formátovat text kurzivou

- Uživatel může formátovat označený úsek kurzivou
  - *Popis: Uživatel chce mít část textu formátovanou kurzívou. Systém by měl umožnit označenému úseku textu nastavit skloněné písmo.*
- Uživatel může zapnout trvalé kurzívou
  - *Popis.: Uživatel chce, aby vkládaný text byl přímo kurzivou a nemusel jej později označovat a nastavit kurzívu dodatečně. Systém by měl obsahovat tlačítko, které sepne psaní kurzívou, tak aby každý nově vložený znak byl kurzívy.*

Editor umí formátovat text podtrženým písmem

- Uživatel může formátovat označený úsek podtrženým písmem
  - *Popis: Uživatel chce mít část textu podtrženou. Systém by měl umožnit označenému úseku textu nastavit podtržení.*
- Uživatel může zapnout trvalé psaní podtrženým textem
  - *Popis.: Uživatel chce, aby vkládaný text byl podtržený přímo a nemusel jej později označovat a nastavit podržení dodatečně. Systém by měl obsahovat tlačítko, které sepne psaní s podtržením, tak aby každý nově vložený znak byl podtržený.*

Editor umí nastavit barvu písma

- Uživatel může změnit barvu písma označeného úseku.
  - *Popis: Uživatel chce mít část textu jinou barvou písma. Systém by měl umožnit označenému úseku textu nastavit uživatelem zvolenou barvu písma.*
- Uživatel může zapnout trvalé psaní zvolenou barvou
  - *Popis.: Uživatel chce, aby vkládaný text byl psán přímo zvolenou barvou písma a nemusel jej později označovat a nastavit barvu písma dodatečně. Systém by měl obsahovat tlačítko, které sepne psaní zvolenou barvou, tak aby každý nově vložený znak byl se zvolenou barvou písma.*

Editor umí nastavit barvu podbarvení písma

- Uživatel může změnit barvu podbarvení označeného úseku.
  - *Popis: Uživatel chce mít část textu zvýrazněnou jinou barvou. Systém by měl umožnit označenému úseku textu nastavit podbarvení uživatelem zvolenou barvou.*
- Uživatel může zapnout trvalé podbarvení písma zvolenou barvou
  - *Popis.: Uživatel chce, aby vkládaný text byl podbarven přímo zvolenou barvou a nemusel jej později označovat a nastavit barvu podbarvení dodatečně. Systém by měl obsahovat tlačítko, které sepne podbarvení zvolenou barvou, tak aby každý nově vložený znak byl se zvoleným podbarvením.*

Editor umí nastavit velikost písma

- Uživatel může změnit velikost písma označeného úseku.
  - *Popis: Uživatel chce mít část textu jinou velikostí písma. Systém by měl umožnit označenému úseku textu nastavit uživatelem zvolenou velikost písma.*
- Uživatel může zapnout trvalé psaní zvolenou velikostí písma
  - *Popis.: Uživatel chce, aby vkládaný text byl psán přímo zvolenou velikostí písma a nemusel později označovat text a nastavit velikost písma dodatečně. Systém by měl obsahovat tlačítko, které nastaví zvolenou velikost písma, tak aby každý nově vložený znak byl ve zvolené velikosti.*

Editor umí kombinovat různé formátování textu

- Uživatel může nastavit označenému úseku textu současně více formátování
  - *Popis: Uživatel chce mít například část textu podtrženou a červeným písmem. Editor by měl umožnit nastavení více kombinací různých formátování současně.*
- Uživatel může zapnout trvalé psaní s různou kombinací formátování
  - *Popis.: Uživatel chce psát přímo se zvolenou kombinací formátování. Uživatel navolí požadované formátování a každý nový znak bude zobrazen s tímto nastavením.*

4.2 Editor SpreadsheetML

Editor umí nastavit tučné písmo buňce

- Uživatel může označit buňku a nastavit ji tučné písmo
  - *Popis: Uživatel chce mít buňku zvýrazněnou tučným písmem. Systém by měl umožnit označené buňce nastavit tučné písmo.*

Editor umí nastavit kurzivé písmo buňce

- Uživatel může označit buňku a nastavit ji kurzivé písmo
  - *Popis: Uživatel chce mít buňku zvýrazněnou kurzivým písmem. Systém by měl umožnit označené buňce nastavit kurzivé písmo.*

Editor umí nastavit podtržené písmo buňce

- Uživatel může označit buňku a nastavit ji kurzivé písmo
  - *Popis: Uživatel chce mít text v buňce podtrženým písmem. Systém by měl umožnit označené buňce nastavit podtržené písmo.*

Editor umí nastavit barvu písma buňky

- Uživatel může označit buňku a nastavit jí barvu písma
  - *Popis: Uživatel chce mít text v buňce zvolenou barvou písmem. Systém by měl umožnit označené buňce nastavit uživatelem zvolenou barvu písma.*

Editor umí nastavit barvu pozadí buňky

- Uživatel může označit buňku a nastavit jí barvu pozadí
  - *Popis: Uživatel chce mít pozadí buňky zvolenou barvou. Systém by měl umožnit označené buňce nastavit uživatelem zvolenou barvu pozadí.*

Editor umí nastavit velikost písma v buňce

- Uživatel může označit buňku a nastavit ji velikost písma.
    - *Popis: Uživatel chce mít text v buňce zvolenou velikostí písma. Systém by měl umožnit označené buňce nastavit uživatelem zvolenou velikost písma.*

Editor umí kombinovat různé formátování textu

- Uživatel může nastavit označenému buňce současně více formátování
    - *Popis: Uživatel chce mít buňku například podtrženou a červeným písmem. Editor by měl umožnit nastavení více kombinací různých formátování současně.*

Editor umí použít vzorec pro sumu

- Uživatel může do buňky vložit funkci suma.
    - *Popis: Uživatel potřebuje udělat sumu hodnot v rozsahu buněk. Vybere si buňku, která má obsahovat sumu a pomocí tlačítka vloží funkci suma. Systém mu zobrazí možnost zadat požadovaný rozsah adres buněk*

5. Rozsah kvality

5.1 Lokalizovatelnost

- Systém může být lokalizován do více jazyků
- Názvy funkcí, data a časové hodnoty musí být ukládány ve formátu definovaném ve standardu Open XML.

5.2 Použitelnost

- Systém je ovládán skrze intuitivní rozhraní.

5.3 Dostupnost

- Aplikace musí být použitelná 24/7

5.4 Integrita dat

- Soubory používané aplikací musí splňovat specifikace Open XML

6. Přednosti a priority

Priorita	1	3	5
Důležitost	**Vysoká**	**Střední**	**Nízká**

- **Priorita 1**
  - Otevření a zobrazení dokumentu WordProcessingML
  - Otevření a zobrazení dokumentu SpreadsheetML
  - Uložení dokumentu WordProcessingML
  - Uložení dokumentu SpreadsheetML

- **Priorita 2**
  - Nastavení formátování označeného textu v editoru WordProcessingML
  - Nastavení formátování nově vkládaného textu v editoru WordProcessingML
  - Nastavení formátování označené buňky v editoru SpreadsheetML

- **Priorita 3**
  - Použití funkce suma v editoru SpreadsheetML

7. Ostatní požadavky na systém

7.1 Systémové požadavky

- Operační systém Google Android 2.1 nebo novější

7.2 Požadavky na prostředí

- Hardware: Procesor 500Mhz, dotykový displej, 256MB RAM, 30MB interní paměti, externí uložiště s volnou kapacitou nejméně 10MB.

Kancelářský balík FreeOffice

FreeOffice

Software Architecture Document

Příloha č. 2

Verze <1.2>

Historie revizí

Datum	Verze	Popis	Autor
14. 01. 2010	1.0		Bc. Pavel Bednář
24. 04. 2011	1.1		Bc. Pavel Bednář
01. 05. 2011	1.2		Bc. Pavel Bednář

Obsah

1. Úvod

1.1 Účel dokumentu

Tento dokument poskytuje komplexní přehled architektury aplikace, s použitím množství různých architektonických pohledů na zachycení různých aspektů systému. Je určena k zachycení a zprostředkovat významných architektonických rozhodnutí, které byly provedeny v aplikaci.

1.2 Rozsah

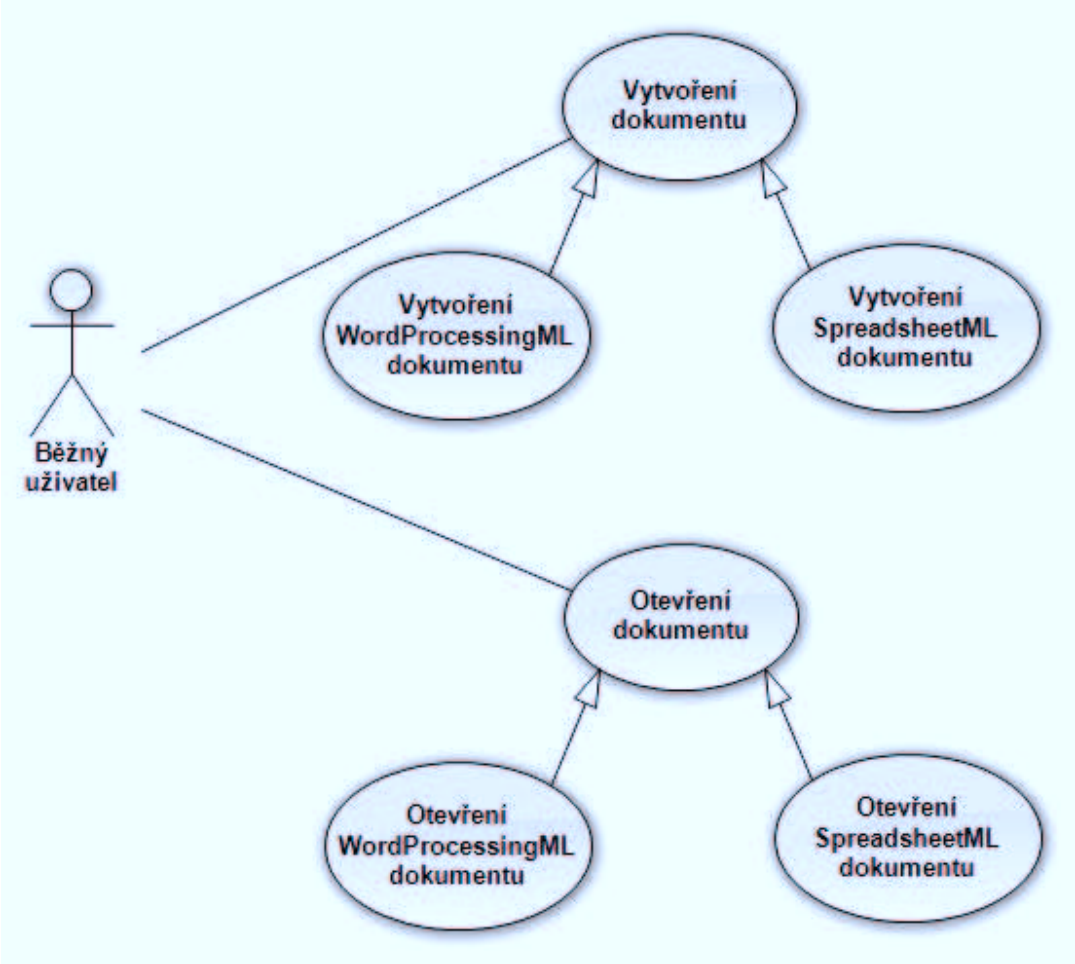
Tento dokument popisuje pohled na architekturu aplikace FreeOffice. Aplikace slouží jako jednoduchý editor formátu Open Office XML. Tento dokument byl vygenerován na základě dokumentu vize a dalších materiálů použitých při návrhu specifikace.

2. Architektura

Je zde několik klíčových požadavků a systémových omezení, které mají významný vliv na architekturu.

- Systém musí dodržovat standart formátů Office Open XML
- Aplikace musí být schopna zobrazit obsah dokumentů dodržující standart Office Open XML
- Dokumenty vytvořené aplikací musí být zpracovatelné pro jiné aplikace Office Open XML

3. Případy užití



3.1 Vytvoření dokumentu

Minimální záruky: Nic.
Maximální záruky: Systém uloží dokument na paměťovou kartu
Vstupní podmínka: Nic.
Výstupní podmínka: Systém smaže dočasnou složku po kompresi balíčku

Popis:

1. Systém zobrazí uživateli hlavní nabídku
2. Uživatel stiskne tlačítko vytvořit
3. Systém zobrazí nabídku s volbou, zda vytvořit nový dokument nebo sešit
4. Uživatel vybere jednu z možností
5. Uživatel vybere volbu dokument
6. Systém vykreslí editor pro práci s dokumenty.
7. Uživatel stiskne tlačítko menu.
8. Systém zobrazí menu.
9. Uživatel zvolí položku uložit.
10. Systém zobrazí dialog pro zadání jména souboru.
11. Uživatel zvolí jméno souboru.
12. Systém uloží dokument (případ užití 3.1.1).

Rozšíření:

5a Uživatel vybere volbu sešit

1. Systém vykreslí editor pro práci s dokumenty.
2. Uživatel stiskne tlačítko menu.
3. Systém zobrazí menu.
4. Uživatel zvolí položku uložit.
5. Systém zobrazí dialog pro zadání jména souboru.
6. Uživatel zvolí jméno souboru.
7. Systém uloží dokument (případ užití 3.1.2).

3.1.1 Uložení WordProcessingML dokumentu

Minimální záruky: Nic.
Maximální záruky: Systém uloží dokument na paměťovou kartu.
Vstupní podmínka: Uživatel zvolil název souboru.
Výstupní podmínka: Systém smaže dočasnou složku po kompresi balíčku.

Popis:

1. Systém načte text z editoru.
2. Systém vytvoří dočasnou složku na paměťové kartě.
3. Systém vytvoří v dočasné složce soubor *document.xml.*
4. Systém vytvoří v dočasné složce soubor *[Content_Types].xml.*
5. Systém vytvoří v dočasné složce soubor *_rels/.rels.*
6. Systém zabalí dočasnou složku.
7. Systém smaže dočasnou složku.

3.1.2 Uložení SpreadsheetML dokumentu

Minimální záruky: Nic.
Maximální záruky: Systém uloží dokument na paměťovou kartu.
Vstupní podmínka: Uživatel zvolil název souboru.
Výstupní podmínka: Systém smaže dočasnou složku po kompresi balíčku.

Popis:

1. Systém načte kolekci buněk z editoru.
2. Systém vytvoří dočasnou složku na paměťové kartě.
3. Systém vytvoří v dočasné složce soubor *workbook.xml.*
4. Systém vytvoří v dočasné složce soubor *[Content_Types].xml.*
5. Systém vytvoří v dočasné složce soubor *worksheet1.xml.*
6. Systém vytvoří v dočasné složce soubor *_rels/.rels.*
7. Systém vytvoří v dočasné složce soubor *_rels/workbook.xml.rels.*
8. Systém zabalí dočasnou složku.
9. Systém smaže dočasnou složku.

3.2 Otevření dokumentu

Minimální záruky: Nic.
Maximální záruky: Systém uloží dokument na paměťovou kartu
Vstupní podmínka: Nic.
Výstupní podmínka: Systém smaže dočasnou složku po kompresi balíčku

Popis:

1. Systém zobrazí uživateli hlavní nabídku
2. Uživatel stiskne tlačítko otevřít
3. Systém zobrazí dialog pro zadání cesty k dokumentu
4. Systém ověří, zda soubor existuje
    a. Pokud soubor neexistuje, systém zobrazí upozornění a opakuje se od kroku 3.
5. Systém rozhodne, jestli se jedná o dokument nebo sešit
    a. Systém zobrazí dokument (případ užití 3.2.1)
    b. Systém zobrazí sešit (případ užití 0)

3.2.1 Otevření WordProcessingML dokumentu

Minimální záruky: Nic.
Maximální záruky: Systém zobrazí dokument v uživatelském rozhraní.
Vstupní podmínka: Uživatel zvolil název souboru.
Výstupní podmínka: Nic.

Popis:

1. Systém vytvoří dočasnou složku.
2. Systém rozbalí uživatelův soubor do dočasné složky
3. Systém otevře soubor *_rels/.rels* a podívá se, kde má hledat *document.xml*.
4. Systém otevře soubor *_rels/document.xml.rels* a podívá se, kde má hledat *styles.xml*.
5. Systém zpracuje a zformátuje data a zobrazí je do editoru
6. Systém smaže dočasnou složku.

3.2.2 Uložení SpreadsheetML dokumentu

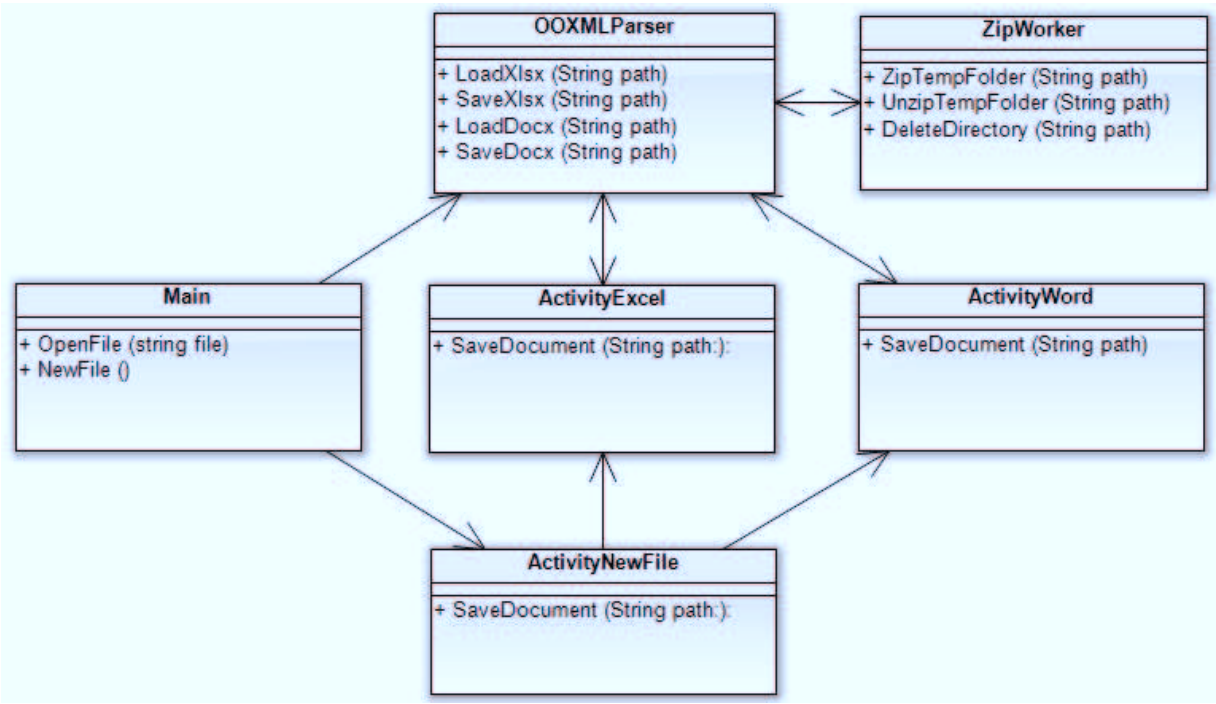
Minimální záruky: Nic.
Maximální záruky: Systém zobrazí dokument v uživatelském rozhraní.
Vstupní podmínka: Uživatel zvolil název souboru.
Výstupní podmínka: Nic.

Popis:

1. Systém vytvoří dočasnou složku.
2. Systém rozbalí uživatelův soubor do dočasné složky
3. Systém otevře soubor *_rels/.rels* a podívá se, kde má hledat *workbook.xml*.
4. Systém otevře soubor *_rels/wokbook.xml.rels* a podívá se, kde má hledat listy sešitu.
5. Systém otevře soubor *_rels/wokbook.xml.rels* a podívá se, kde má hledat tabulku sdílených řetězců *sharedStrings.xml*.
6. Systém načte první list, který je definovaný v sešitě. A na základě adres buněk a sdílených řetězců zobrazí data.
7. Systém smaže dočasnou složku.

4. Logický pohled

Popisuje nejdůležitější třídy, jejich organizací uvnitř balíčků a subsystémů, a jejich organizaci do vrstev. Také popisuje nejdůležitější use-case realizací, například dynamické aspekty architektury. Třídní diagramy mohou být zahrnuty pro ilustraci vztahů mezi architektonicky významnými třídami, subsystémy, balíky a vrstvami.

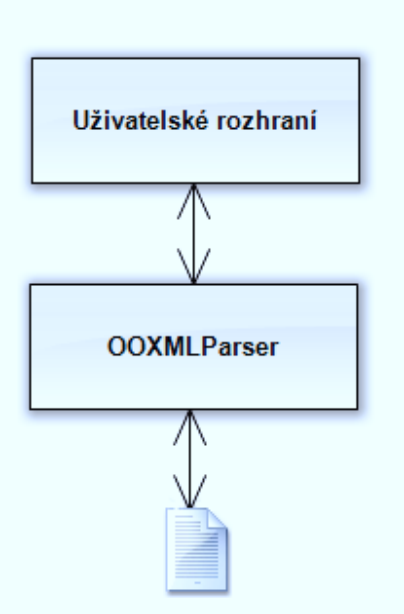


5. Nasazení aplikace

Aplikace bude dostupná ke stažení z webové stránky nebo časem i přes službu Android Marketu. Aplikace poběží na samostatném zařízení a nebude komunikovat s žádným jiným systémem či zařízením.

6. Implementační pohled

Popisuje, jak by měl být systém implementován. Systém bude pracovat v podstatě na dvou vrstvách. Spodní vrstva OOXMLParser bude mít na starosti zpracování souborů Open XML. Data pro ukládání bude získávat z uživatelského rozhraní. Tyto data zpracuje a uloží do souboru. Při otvírání souboru OOXMLParser zpracuje soubor a zpracovaná data poskytne uživatelskému rozhraní pro zobrazení.



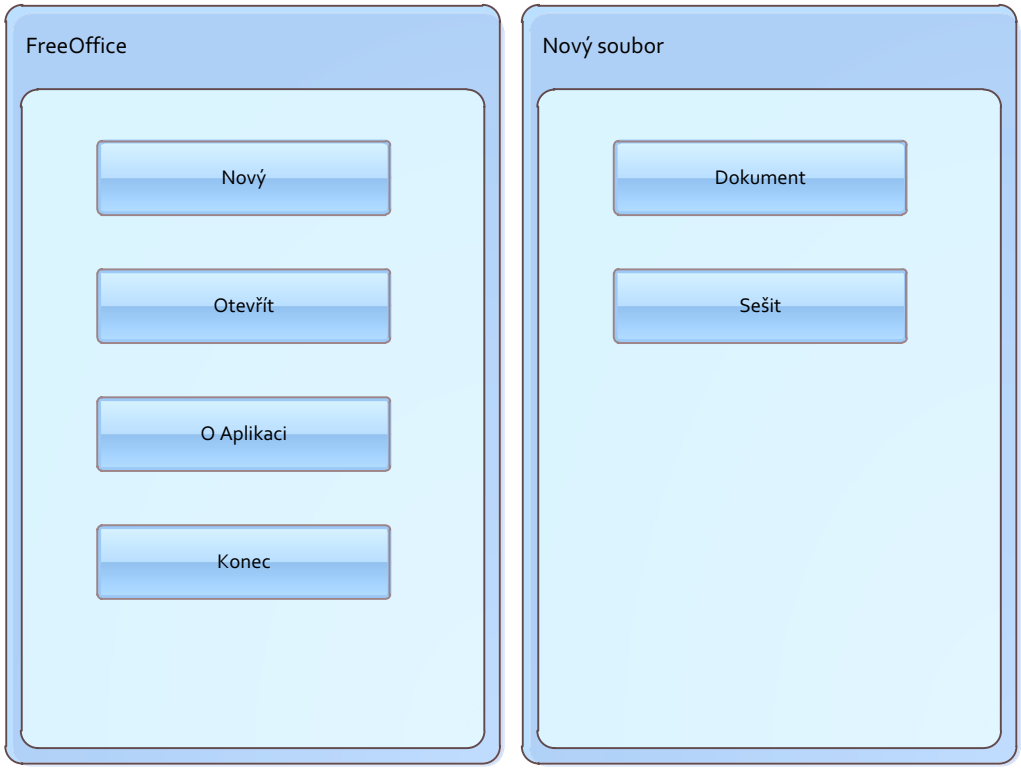
Obrázek 1 Architektura aplikace

7. Uživatelské rozhraní

Tato část popisuje, jak by měli vypadat některé důležité formuláře aplikace. Jde pouze o návrh, není to finální podoba. Pouze návrh, jak by měli formuláře vypadat.

7.1 Hlavní menu

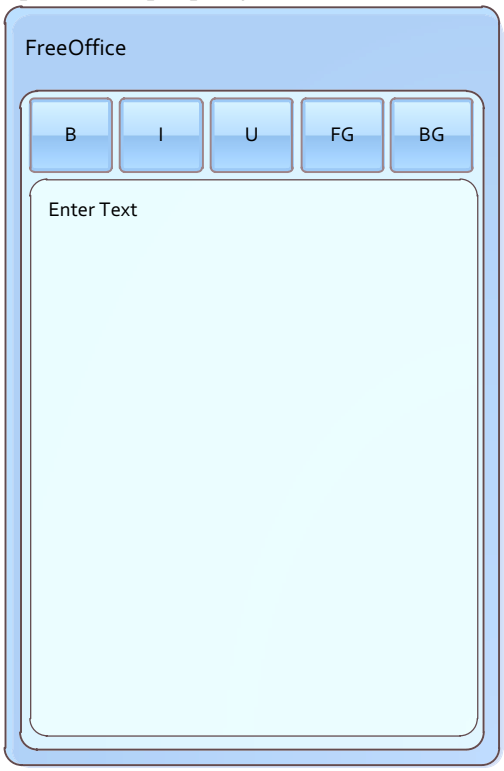
Úvodní obrazovka, která se zobrazí uživateli při spuštění aplikace. Pokud si zvolí, že chce vytvořit nový soubor, tak se zobrazí další obrazovka, kde si zvolí, jestli chce vytvořit dokument nebo sešit.



Obrázek 2 Hlavní menu a menu pro tvorbu nového souboru

7.2 Editor WordProcessingML

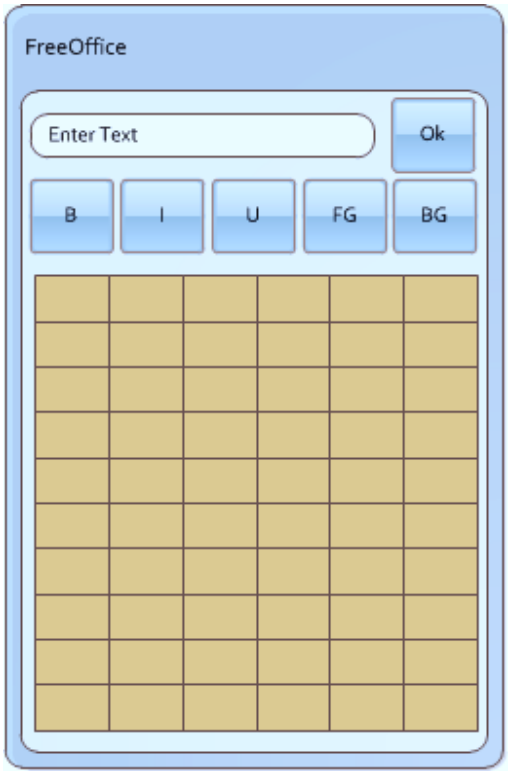
Návrh uživatelského rozhraní pro zpracování textových dokumentů. Obsahuje nástrojovou lištu pro rychlé nastavení formátování a textové pole, které podporuje formátování svého obsahu.



Obrázek 3 Editor WordProcessingML

7.3 Editor SpreadsheetML

Návrh uživatelského rozhraní pro zpracování sešitu. Obsahuje nástrojovou lištu pro rychlé nastavení formátování. A lištu pro zadávání hodnot do buněk.



Obrázek 4 Editor pro SpreadsheetML

8. Rozsah kvality

8.1 Lokalizovatelnost

- Systém může být lokalizován do více jazyků
- Názvy funkcí, data a časové hodnoty musí být ukládány ve formátu definovaném ve standardu Open XML.

8.2 Použitelnost

- Systém je ovládán skrze intuitivní rozhraní.

8.3 Dostupnost

- Aplikace musí být použitelná 24/7

8.4 Integrita dat

- Soubory používané aplikací musí splňovat specifikace Open XML

8.5 Systémové požadavky

- Operační systém Google Android 2.1 nebo novější

8.6 Požadavky na prostředí

- Hardware: Procesor 500Mhz, dotykový displej, 256MB RAM, 30MB interní paměti, externí uložiště s volnou kapacitou nejméně 10MB.